A Manual for Easy Nonlinear Parameter Estimation in Fishery Research with Interactive Microcomputer Programs

Jon Schnute

Department of Fisheries and Oceans
Fisheries Research Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

December 1982

Canadian Technical Report of Fisheries and Aquatic Sciences No. 1140

Canadian Technical Report of Fisheries and Aquatic Sciences

These reports contain scientific and technical information that represents an important contribution to existing knowledge but which for some reason may not be appropriate for primary scientific (i.e. *Journal*) publication. Technical Reports are directed primarily towards a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries management, technology and development, ocean sciences, and aquatic environments relevant to Canada.

Technical Reports may be cited as fuli publications. The correct citation appears above the abstract of each report. Each report will be abstracted in *Aquatic Sciences and Fisheries Abstracts* and will be indexed annually in the Department's index to scientific and technical publications.

Numbers 1-456 in this series were issued as Technical Reports of the Fisheries Research Board of Canada. Numbers 457-714 were issued as Department of the Environment, Fisheries and Marine Service, Research and Development Directorate Technical Reports. Numbers 715-924 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Technical Reports. The current series name was changed with report number 925.

Details on the availability of Technical Reports in hard copy may be obtained from the issuing establishment indicated on the front cover.

Rapport technique canadien des sciences halieutiques et aquatiques

Ces rapports contiennent des renseignements scientifiques et techniques qui constituent une contribution importante aux connaissances actuelles mais qui, pour une raison ou pour une autre, ne semblent pas appropriés pour la publication dans un journal scientifique. Il n'y a aucune restriction quant au sujet, de fait, la série reflète la vaste gamme des intérêts et des politiques du Ministère des Pêches et des Océans, notamment gestion des pêches, techniques et développement, sciences océaniques et environnements aquatiques, au Canada.

Les Rapports techniques peuvent être considérés comme des publications complètes. Le titre exact paraîtra au haut du résumé de chaque rapport, qui sera publié dans la revue *Aquatic Sciences and Fisheries Abstracts* et qui figurera dans l'index annuel des publications scientifiques et techniques du Ministère.

Les numéros 1-456 de cette série ont été publiés à titre de Rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457-714, à titre de Rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715-924 ont été publiés à titre de Rapports techniques du Service des pêches et de la mer, Ministère des Pêches et de l'Environnement. Le nom de la série a été modifié à partir du numéro 925.

La page couverture porte le nom de l'établissement auteur où l'on peut se procurer les rapports sous couverture cartonnée.

Canadian Technical Report of Fisheries

and Aquatic Sciences No. 1140

December 1982

A MANUAL FOR EASY NONLINEAR PARAMETER ESTIMATION IN FISHERY RESEARCH WITH INTERACTIVE MICROCOMPUTER PROGRAMS

by

Jon Schnute

Department of Fisheries and Oceans

Fisheries Research Branch

Pacific Biological Station

Nanaimo, British Columbia V9R 5K6

(c) Minister of Supply and Services Canada 1982

Cat No. Fs 97-6/1140 ISSN 0706-6457

TABLE OF CONTENTS

	Page
List of tables	vi
List of figures	vi
List of computer listings	vii
Abstract	viii
Résumé	ix
Preface	x
Preface to the second edition	xii
1. Introduction	1
2. The simplex search method	7
Reflection	9
Extension	9
Contraction	9
Reduction	9
3. The user program	10
Variables (Table 3.1)	11
Auxiliary parameters	12
Tasks (Figure 3.1)	13
First level rules	14
Second level rules	15
TEMPLATE	15
TRY	17
4. MICRO SIMPLEX	21
Input data	
1. Initial point and step	21
2. Required standard deviation	24
3. Restart step reduction	24
4. Maximum number of calls	24
5. Console display code	24
6. Printer display code	24
7. Convergence check code	26
Tuning summary	30
Simplex algorithm alone	30
5. PROFILE	30
Data entry	32

P, E, S, L, or Q 35
Prefix PRO. 35
PROTECT 35

6. PREDICT 36
PROTECT 36
Data entry 36
Prefix PRED. 36

7. Problems to watch for 36
7.1 Multiple minima 39
MULTIMIN 39
7.2 Choosing the right parameters 42
GROWTH 42
GROW PARS 42
REGRESS 43

8. Additional possibilities 43
8.1 Arbitrary information functions 43
Bayes estimates 43
8.2 Some parameters held fixed 43
Analysis of variance 44
8.3 Some parameters linear 46
MINV1, MINV2 46
8.4 Constrained problems 47
8.5 Nonlinear equations 48

References 49
Appendix A. Technical requirements and suggestions . . . 50
A.1 General requirements 50
Applesoft Toolkit APA 50
RENUMBER 50
A.2 Graphics memory protection 50
PROTECT 50
HELLO 50
A.3 Compressed programs 50
Dakin 5 CRUNCHER 51
A.4 Hard copy graphics 51
Computer Station Paper Tiger software 51
A.5 Vector list subroutine 51
A.6 Data files 51
COMPUSHOP DATABASE 51
TEMPLATE 51
Applesoft Toolkit EDIT 51
LENGTH DATA, WEIGHT DATA 51

Appendix B. Version 1.1 files 52

Appendix C. Worked example 53

Page

Appendix D. Program listings 55
Dakin 5 LISTER 55
Dakin 5 VARIABLE CROSS REFERENCE 55
GROWTH 56
GROW PARS 63
HELLO 68
MICRO SIMPLEX 70
MINV1 83
MINV2 85
MULTIMIN 87
PREDICT 90
PROFILE 97
PROTECT 104
REGRESS 106
TEMPLATE 110
TRY 113

TABLES

Table		Page
3.1	User program special variable names	11
4.1	Minimum test, example	25
4.2	Simplex algorithm I/O variable names	31

FIGURES

Figure		
1.1	Two predicted curves for Ricker's cisco length data (PREDICT example)	4
1.2	Variations in the sum of squares S_1 for Ricker's cisco length data (PROFILE example)	6
2.1	Simplex search iterations	8
3.1	User program tasks	13
7.1	Profile showing multiple minima (MULTIMIN example)	41

COMPUTER LISTINGS

Listing		Page
3.1	User program for the data analysis in Section 1 (Second level example)	16
3.2	User program for normal parameters M and S (First level example)	18
3.3	Trial run of Listing 3.1 (TRY example with "F" option)	19
3.4	Trial run of Listing 3.1 (TRY example with "P" option)	20
4.1	MICRO SIMPLEX input for Listing 3.1	22
4.2	MICRO SIMPLEX output from Listing 3.1 (detailed)	27
4.3	MICRO SIMPLEX output from Listing 3.1 (coarse)	28
5.1	PROFILE input/output for Listing 3.1 and Figure 1.2	33
6.1	PREDICT input/output for Listing 3.1 and Figure 1.1	37
7.1	PROFILE descriptions for Figure 7.1	40
8.1	MICRO SIMPLEX run with Listing 3.1 when one parameter is fixed	45

ABSTRACT

Schnute, Jon. 1982. A manual for easy nonlinear parameter estimation in fishery research with interactive microcomputer programs. Can. Tech. Rep. Fish. Aquat. Sci. 1140: xvi + 115 p.

This manual describes in detail how to solve many practical problems encountered in nonlinear parameter estimation. In addition, it presents software to aid the user with three tasks: (1) finding optimal parameter estimates, (2) plotting observations and model predictions, and (3) displaying graphically the variation in likelihood (or sum of squares) when the parameters are varied from their optimal estimates. This software is coded in BASIC for the Apple II microcomputer, and it is available on a suitable 5 1/4" diskette. In many cases, the user can adapt the general software here to his or her particular problem by adding just a few lines of BASIC code.

The simplex method of searching for a function minimum lies at the heart of the discussion here. This manual describes the method completely. Although a simplex search is known to be less efficient of compute time than derivative-based methods, it has the considerable advantage of minimizing human time required for coding a particular problem. The manual places great emphasis on ease of program development, even at the expense of computer time. It also shows how to adjust the simplex method for optimal efficiency and how to apply it in other contexts besides nonlinear estimation. The discussion throughout is illustrated with numerous examples from fisheries literature, although the methods have obvious application in many fields.

The broad purpose of this manual is to make the reader as comfortable with the process of nonlinear estimation as with the much simpler standard procedure of linear regression.

Key words: parametric models, parameter estimation, nonlinear estimation, minimization, function minimization, simplex search, microcomputers, Apple II microcomputer, BASIC, likelihood, maximum likelihood, fisheries models.

RESUME

Schnute, Jon. 1982. A manual for easy nonlinear parameter estimation in fishery research with interactive microcomputer programs. Can. Tech. Rep. Fish. Aquat. Sci. 1140: xvi + 115 p.

Le présent manuel décrit en détail la manière de résoudre plusieurs problèmes pratiques recontrés dans l'estimation non linéaire des paramètres. De plus, il présente le logiciel nécessaire pour aider l'usager dans les trois tâches suivantes: (1) la recherche des estimations optimales de paramètres, (2) le traçage des observations et des prédictions de modèles, et (3) la visualisation graphique de la variation de la possibilité (ou somme des carrés) quand les paramètres diffèrent de leurs estimations optimales. Ce logiciel est codé en BASIC pour le micro-ordinateur Apple II et il est disponible sur disquette appropriée de 5 1/4". Dans plusieurs cas, l'usager peut adapter le logiciel général à ses besoins particuliers en ajoutant quelques lignes en BASIC.

Cette méthode de transmission unidirectionnelle pour la recherche d'un minimum pour la fonction constitue la partie principale de la discussion. Toute la méthode est décrite dans le manuel. Quoique la recherche par transmission unidirectionnelle utilise moins efficacement le temps sur ordinateur que les méthodes à base dérivée, elle a l'avantage considérable de minimiser le temps qu'une personne doit passer à coder un problème. Le manuel vise surtout à faciliter l'élaboration de programmes, même aux dépens du temps sur ordinateur. Il montre aussi comment ajuster la méthode par transmission unidirectionnelle pour obtenir une efficacité maximale et comment l'appliquer dans des contextes autres que les estimations non linéaires. Toute la discussion est illustrée de nombreux exemples tirés d'ouvrages sur les pêches, mais les méthodes peuvent évidemment s'appliquer à d'autres domaines.

Le but général de ce manuel est de familiariser le lecteur avec le procédé d'estimation non linéaire autant qu'avec la procédure classique beaucoup plus simple de régression linéaire.

Mots-clés: modèles paramétriques, estimation de paramètre, estimation non linéaire, minimisation de fonction, recherche par transmission unidirectionnelle, micro-ordinateurs, micro-ordinateur Apple II, BASIC, probabilité, probabilité maximale, modèles de pêche.

PREFACE

Fisheries literature in recent years includes an increasing number of analyses based on nonlinear parametric models. Some of these papers have been authored or co-authored by myself. Each time I've participated in such a paper, I've been left wondering just how easily readers might be able to put the ideas into practice. For linear models, one can give formulas for the parameter estimates which translate readily to a computer program. Nonlinear parametric models require the user to program an information function, such as the likelihood or a sum of squared model errors, which must then be optimized. If the reader feels comfortable with this process, then it's enough to specify the appropriate information function. If not, then he or she may be left feeling that, no matter how interesting the analysis looks, it could not actually be attempted without an enormous effort.

Current developments in microcomputing technology pose some remarkable possibilities for implementing the required optimization methods. While microcomputers are hardly the perfect environment for a large numerical project like nonlinear estimation, they can perform acceptably on many of the small- to medium-sized problems which commonly occur in fishery data analysis. The ease of program development in BASIC helps compensate for some of the inconveniences, like slow operation. Also, these small systems have highly portable hardware and software. In the Canadian Department of Fisheries and Oceans, Pacific Region, for example, there are now several Apple II microcomputers. This report, together with one diskette, makes it possible for any user with access to an Apple II to take immediate advantage of all the programs described here. I particularly hope that the programs here might prove useful to fishery management in developing countries where large computers simply aren't available.

The intent of this report is to present a tool which makes it almost as easy for the practitioner to estimate parameters in moderate-sized nonlinear models as in linear ones, where ordinary regression gives the answer in a single step. I propose to achieve this by (1) tailoring the tool so that it does as much of the practitioner's work as possible, (2) rendering its operation visible enough so that he or she can see clearly how the tool performs, and (3) providing easy graphics display to assess the outcome. Essentially, the user needs to supply only a program to calculate the information function, such as (but not necessarily) a sum of squares of model errors. Extra conditions, such as parameter constraints or Bayesian priors, can be added easily. The tool readily allows selected parameters to be fixed at specified values.

The underlying methods employed here certainly are not new. General techniques for function minimization have been studied extensively, and a variety good algorithms exist in FORTRAN to do the job effectively on a large

computer. This report employs the simplex search method originally proposed by Nelder and Mead (1965). The BASIC algorithm developed here is somewhat more flexible and much more interactive than its commonly referenced FORTRAN counterpart (O'Neill, 1971). The current version is based on my practical experience over about a year and a half. This does not imply that the program is optimal or totally error-free, and I would gratefully welcome any discussion of problems or suggestions for improvement.

I became involved in this project for some rather peculiar reasons. The Pacific Biological Station, where I have worked since 1976, does not (as of this writing in early 1981) have a full-scale interactive computing system. Since interactive program development and use is almost essential for efficient nonlinear estimation, I began my work on the only available resource: a Data General Nova 2 microcomputer which is operated exclusively in BASIC with about 9000 bytes in central memory available to the user for programs and data. The simplex method was the best available for tailoring to these contraints, and the programs here (wihout graphics) were originally developed for the Nova 2. Later, an Apple II microcomputer became available with an effective user space of 36000 bytes. In the context of computing at the Biological Station, this felt like a very large workspace, and the programs were enlarged to their present form, along with graphics routines to exploit the Apple's handy capabilities in that regard.

In some cases, especially when the model involves more than five parameters, the programs here may run quite slowly. Literature on the simplex algorithm commonly states that it performs poorly with more than four parameters, but I have used it successfully with up to 15. It's partly a matter of patience. Sometimes the computer has to be left alone for half-an-hour, or for several hours, or even overnight, while it searches for optimal parameter estimates. In a research context, the ease of program development is often much more important than computer run-time. If a model takes just a few minutes to program, then it can easily be tried, even if it means letting the computer work for a while. On the other hand, if a model needs hours or days of programming, then it may not be tried, even if the computer could do the work in a few minutes.

This report describes programs possible on the Apple II Plus with 48K memory and with the MICRO SIMPLEX system diskette, Version 1.1. Further details on system requirements appear in Appendix A. Appendix B lists the files available on the version 1.1 diskette, together with the sections of this report where each file is discussed. I anticipate future enhanced versions, and possibly even a complete FORTRAN counterpart. Naturally, with these possibilities in mind, I emphasize again that I would welcome all user corrections, suggestions, and proposals for future software design.

Jon Schnute
Pacific Biological Station

April, 1981

PREFACE TO THE SECOND EDITION

Strictly speaking, this document never appeared in a first edition, except for some preprints given to a few trial users. A paper of mine (Schnute 1981) refers to this manual as "Can. Spec. Publ. Fish. Aquat. Sci. 59 (in press)". Indeed, although the manual was originally written for the Technical Report series, the editor requested that I submit it instead to the Special Publications series. However, when the editor decided to seek a review, the referee did not recommend publication without significant changes and additions. As I explain below, I agree with some of the criticism and wish I had time to produce a manual and software that dealt with all of it. Unfortunately, research on specific problems of fishery management continually takes priority. Meanwhile, almost a year and a half has gone by since the first edition was completed. I have had dozens of requests from several countries (including APPLE owners in remote places) for this paper based only on its reference in Schnute (1981), and I continue to find the programs of considerable use. I believe many practitioners may also find value in the methods here, so I have decided to make them available in the Technical Report series without further delay.

In order to clarify this manual's purpose and limitations, I'd like to address the referee's main comments here. Readers interested in making active use of this material will do well to take a few minutes to follow the discussion below. In each case I give a short paraphrase of the referee's comment (underlined), together with my reply.

1. <u>This manual describes a particular method of function minimization, but that is a very small subset of the problem of model selection and fitting</u>.

I agree. The whole point of this manual is to document a technology for getting parameter estimates easily. I assume that the user has defined the problem properly and simply wants to find the estimates. I'm trying to make this job almost as easy as in the case of linear regression, where a formula gives the answer in one step. Just as models which depend linearly on their parameters can be wrongly applied, so too can models with parameters which enter nonlinearly. The matter of selecting apprpopriate models isn't addressed here, except for some limited discussion in section 7.

2. <u>The system should have been written in a higher level language, like APPLE PASCAL (which can be compiled), rather than BASIC</u>.

In principle, that's true. The most annoying feature of BASIC is that it doesn't support true subroutines. All variables are global so that the user continually has to worry about variable names. That problem is reasonably solved here, but it can't be completely circumvented.

Unfortuntely, APPLE PASCAL suffers from serious flaws which render it awkward and even useless in this context. The fatal flaw is that it supports only single (four-byte) precision, and that just isn't enough for

most practical fishery estimation problems. Also, although the operating system is conceptually elegant on the APPLE II, it's slow and awkward to use. Debugging is a serious problem compared with APPLE BASIC. Finally, APPLE PASCAL imposes on the user significant extra costs (at least for the APPLE II) in hardware and software. The material here is primarily intended for the rather large audience of APPLE II users without a particularly enhanced system.

In contrast with PASCAL, APPLE BASIC is implemented with five-byte precision, and that is usually just enough for practical problems. Since the language runs interpretively, it is remarkably easy to debug. Furthermore, there are now several good compilers available on the market, so that programs, after debugging, can be compiled and run at high speed. In fact, the speed of compiled BASIC on the APPLE II is greater than that of so-called "compiled" PASCAL, because PASCAL is actually compiled only to Pseudo-machine code ("P-code"), a slightly higher level language than machine language.

Although the software presented here works well on the APPLE II, it certainly isn't easily transportable. It has been converted to VAX BASIC without too much difficulty, but BASIC varies considerably from system to system. The ideal language for portability is FORTRAN, and plans are underway to put much of the software here into that language. When this is done, it will be made available for the convenience of non-APPLE users.

3. Marquardt's gradient method could have been implemented in exactly the same way as the simplex method, with similar simplicity to the user. Furthermore, the detailed discussion of the simplex method isn't necessary for biologists.

The main convenience of the simplex method is that it is a direct search method which, consequently, doesn't require the user to compute or code derivatives. Gradient-based methods want to know which way is downhill, that is, they depend on a knowledge of the derivatives. When the referee made his comment, he clearly thought of the biologist and the programmer as two different people. The trouble is, in my experience, that the biologist (happily or unhappily) often ends up writing his own programs. I know of instances where biologists have spent days calculating complex derivatives required for gradient-based methods, and this whole labor could have been spared by using the simplex method.

Another nice feature of the simplex method is that it's easy to understand how it works. Gradient-based methods are motivated by some rather complicated analysis, while the simplex method is entirely geometric. It may be that the biologist won't really care to follow the details, but I hope there will be some who are sufficiently curious that they read section 2 and come to terms with Figure 2.1.

The simplex method has the added advantage that it is reasonably robust: sooner or later it usually finds a minimum. Gradient methods sometimes fail for rather mysterious reasons and leave the user helpless in deciding what to do next. This rarely happens with the simplex method, and

users who have taken the trouble to understand how it works will almost always be able to get it to converge. Section 4 includes detailed discussion of this problem; see in particular the "tuning summary" on page 30.

The biggest drawback of the simplex method is that it tends to bog down when the number of parameters becomes large. I have run this software with 15 and even 20 parameters successfully, although sometimes slowly. My philosophy is that I'd rather let the computer spend some extra time than devote my own time to coding derivatives. Still there is no question that eventually a gradient-based method becomes essential. As computing costs diminish, however, so does the need to employ a gradient method.

Readers interested in studying other minimization methods will find excellent readable description in Nash (1979), along with some interesting comparisons among methods. If I had known that Nash's book existed when I wrote this manual originally, I would have cast parts of the coding around his algorithms. His variable metric method (Algorithm 21) would be an excellent candidate for a general gradient method on a small computer. Incidentally, fisheries estimation problems frequently cannot be reduced to least squares, so that practical software must be capable of finding minima of arbitrary functions.

4. No provision is made for calculating the variance-covariance matrix of the parameter estimates.

That's true. Again, the main purpose of this manual is to provide software to find the estimates themselves. However, this additional feature is easy to implement, and software to do so will be included in future work. The referee suggests using Newton's divided difference method to approximate the matrix of second partial derivatives (also called the Hessian) of the negative log-likelihood at its minimum. The inverse of this matrix approximates the covariance matrix of parameters. To the reader who isn't familiar with all this terminology, that may sound like an imposing task, but it requires only a few lines of code, once the function itself has already been coded for use with the simplex algorithm.

5. The matrix inversion routine in section 8.3 is not optimal for this context. Pivoting isn't needed, and the Choleski decomposition should be used.

This is an excellent comment; however, it shouldn't detract from the main point of section 8.3 The idea there is that if some of the parameters enter linearly, they can be found by linear regression as part of each function calculation. This reduces the dimensionality of the problem, and potentially leads to a great improvement in speed. Recent work shows that the simplex method operating in this fashion can be much more efficient than a gradient method operating on the full set of parameters. What the referee disputes is my sledge hammer approach to linear regression. First of all, pivoting isn't necessary in the matrix inversion, as suggested in section 8.3, and secondly the Choleski decomposition is known to be the optimal method of inverting matrices of this kind. The methods described in 8.3 will work; they

just aren't optimal. The book by Nash (1979) cited above gives a compact method (Algorithm 7) for Choleski decomposition.

6. The system doesn't permit weighted residuals in computing the sum of squares.

That's not strictly true. Since the system permits any function to be minimized, in particular it allows for a weighted sum of squares. However, it is true that the module TEMPLATE for the particular case of sums of squares doesn't include the possibility of variable weights. A sensitive reader will note that the data used in the example of section 1 represent mean lengths of fish at various ages. These means come from samples of different sizes, and there is some justification for weighting the residuals differently at each age. The correct procedure depends on the model, that is, on how one defines the population growth curve. I have chosen not to weight the residuals differently, not because that is the only way, but rather to obtain a simple demonstration problem for the software. Once again, this is a manual on the practical problems of estimation, not on the complete science of model building.

7. There is no provision for residual plots, as required for valid model selection.

It is certainly true that residual plots can be useful in deciding whether or not the model is really appropriate to the data. In fact, strange behavior of the residuals can suggest ways to modify the model. I repeat the point made that this manual is directed at the limited problem of estimation. A fully integrated system of software for model building would be valuable, but this document doesn't attempt such an ambitious project.

Concluding Remarks

This manual is intended for biologists who are troubled by the problem of nonlinear parameter estimation. They may feel comfortable with linear regression, but helpless when the parameters enter the model nonlinearly. I hope to demonstrate here that there are straightforward methods of estimation which can be understood completely without sophisticated mathematics and used with ease.

Although estimation can be the most technically difficult problem of model building, it certainly isn't always the part that requires the most thought and skill. As the referee points out, this is only one step of the way. It may be used iteratively: estimate the parameters, see how well the model works, modify it, and estimate again. I have included referee's remarks in some detail here to give the reader at least a glimpse of what else might be involved. If the process of estimation itself can be made easy enough, the practitioner is freed to devote his or her creative powers to the deeper problem of proper model design.

To those who have waited for months to receive this manual, I extend my apologies for the delay.

Jon Schnute
Pacific Biological Station
Nanaimo, B.C. V9R 5K6
Canada

November, 1982

1. INTRODUCTION

It's difficult to go far in the analysis of fishery data without confronting the problem of nonlinear parameter estimation. For example, the familiar von Bertalanffy asymptotic growth model (see Ricker (1975))

$$y = y_\infty(1 - e^{-K(t-t_o)}), \tag{1.1}$$

which expresses length y as a function of time t, leads to a nonlinear estimation problem for y_∞, K, and t_o. This model can be contrasted with the linear or quadratic growth models

$$y = a + bt, \tag{1.2}$$

or

$$y = a + bt + ct^2. \tag{1.3}$$

In (1.2) it is possible to get estimates $\hat{a}$ and $\hat{b}$ for a and b by ordinary regression of y on t. Similarly estimates $\hat{a}$, $\hat{b}$, $\hat{c}$ in (1.3) follow from the regression of y on the variables t and t^2. However, no compact formulas, like those of ordinary regression, will give estimates $\hat{y}_\infty$, $\hat{K}$, and $\hat{t}_o$ in the model (1.1).

This dilemma can be stated more precisely in the following way. Suppose that N known data points (t_i,Y_i), $i = 1, \ldots, N$, are to be used in estimating the parameters for one of the models above. These data determine a sum of squares of differences between observations Y_i and predictions y_i,

$$S = \sum_{i=1}^{N} (Y_i - y_i)^2. \tag{1.4}$$

The predictions y_i depend on the model parameters; consequently, so does S. For example, S takes the special form

$$S_1(y_\infty,K,t_o) = \sum_{i=1}^{N} \left[Y_i - y_\infty(1 - e^{-K(t_i-t_o)}) \right]^2, \tag{1.5}$$

$$S_2(a,b) = \sum_{i=1}^{N} \left[Y_i - a - bt_i \right]^2, \tag{1.6}$$

$$S_3(a,b,c) = \sum_{i=1}^{N} \left[Y_i - a - bt_i - ct_i^2 \right]^2 \tag{1.7}$$

for models (1.1), (1.2), and (1.3) respectively. Ordinary regression gives estimates $\hat{a}$ and $\hat{b}$ which minimize the sum of squares S_2 or similarly, $\hat{a}$, $\hat{b}$, and $\hat{c}$ which minimize S_3. Prescribed formulas for these estimates are possible because models (1.2) and (1.3) depend linearly on their parameters. By

contrast, the parameters in model (1.1) do not enter linearly, and, consequently, it is harder to find estimates $\hat{y}_\infty$, $\hat{K}$, and $\hat{t}_o$ which minimize S_1.

The sum of squares S in (1.4) plays the role of a criterion function for determining optimal parameter estimates. In this case, the deviation between observed length Y_i and predicted length y_i is measured simply by the difference $Y_i - y_i$. It might happen that deviations tend to increase in proportion to length, so that the logarithmic change

$$\log Y_i - \log y_i = \log (Y_i/y_i)$$

would be taken as a measure of deviation. This suggests using the criterion

$$S^* = \sum_{i=1}^{N} [\log (Y_i/y_i)]^2 \tag{1.8}$$

instead of (1.4). For example, model (1.2) would then lead to the problem of minimizing

$$S_2^*(a,b) = \sum_{i=1}^{N} [\log (Y_i/(a + bt_i))]^2. \tag{1.9}$$

Unlike S_2 in (1.6), S_2^* cannot be minimized by linear regression.

All this discussion simply reinforces the point, well-known to many practitioners, that linear regression models are extremely limited. Even apparently linear models like (1.2) can easily lead to nonlinear problems, depending on the error structure, as (1.9) shows. Indeed (1.9) may be more realistic biologically than (1.6) because model errors in fish length may be larger for larger fish. Yet the technical inconvenience of (1.9) may cause the practitioner to reject it in favor of the mathematically simpler (but less realistic) (1.6).

This paper presents a tool for finding parameter estimates appropriate to any user-specified model (like (1.1), (1.2), or (1.3)) and any criterion function (like (1.4) or (1.8)). More generally, the tool can be used to minimize any function of several variables. It can also be used to plot the function as the variables change in certain ways, and it can produce graphs of observed and predicted values. The broad purpose of this tool is to allow the practitioner to find the best statistical fit of a model to his or her data, and to assess how good that fit is. Typically, the user needs to add only a minor bit of BASIC program code to the software given here.

As an example, consider some data given by Ricker (1975, p. 226) for ciscoes (Coregonus artedii) from Vermillion Lake Minnesota. The mean lengths Y in centimeters for ages t from 2 to 11 are given, respectively, as: 172, 210, 241, 265, 280, 289, 294, 302, 299, and 306. Suppose that one wishes to

```
3100 YI = X(1) : K = X(2) : TZ = X(3) : T = XP
3110 YP = YI * (1 - EXP (-K*(T-TZ)))
```

Here the first line identifies the three parameters YI (y-infinity), K, and TZ (t-zero) with the system's parameter vector, labelled X. Also T (time) is identified with the system's x-value for prediction, XP. Then the second line calculates for the system a predicted y-value, YP, on the basis of equation (1.1). That's it; no other user-generated code is required for fitting model (1.1) with the criteria S and S*.

Next it is necessary to enter the length data above into a computer data file. This file needs to have a particular format described in Appendix A. For the purposes of this example, assume that the data are entered. They appear on the system diskette in a file called LENGTH DATA.

Using the two lines of code above and the data file just mentioned, one can now easily find for S and S* the following parameter estimates:

$\hat{y}_\infty$	$\hat{K}$	$\hat{t}_o$	Minimum
310.8	.3623	-.1898	S = 46.3700
311.8	.3523	-.2564	S* = 6.82406×10^{-4}

The entire process of setting up the program and obtaining both these estimates takes about 15 minutes, once the practitioner is familiar with the software. With a reasonable initial guess, the search algorithm calculates S or S* fewer than 200 times to locate the minima shown here. For example, my first wild guess was $y_\infty = 330$, $K = 1$, $t_o = 0$. The algorithm found the first estimate after 158 evaluations of S, and the second estimate after 162 evaluations of S*.

The same user code needed to find parameter estimates can also be used, in connection with another program, to generate pictures of observed data points and predicted curves. This program produced Figure 1.1 which shows (i) the data for this example, (ii) the growth curve corresponding to my first wild guess, and (iii) the predicted growth curve based on the estimates $\hat{y}_\infty$, $\hat{K}$, and $\hat{t}_o$ for the criterion S. Clearly, the observations conform very closely to the curve (iii); that is, the data are not particularly noisy here. Consequently, it is not too surprising that the criteria S and S* both lead to similar parameter estimates. Incidentally, in some cases the practitioner may want to use the graphics program before the estimation program in order to find reasonable initial estimates for the parameters. My own first guess, which misses the data entirely (Fig. 1.1) could certainly be improved this way.

A logical next step in the analysis is to ask how well the estimates

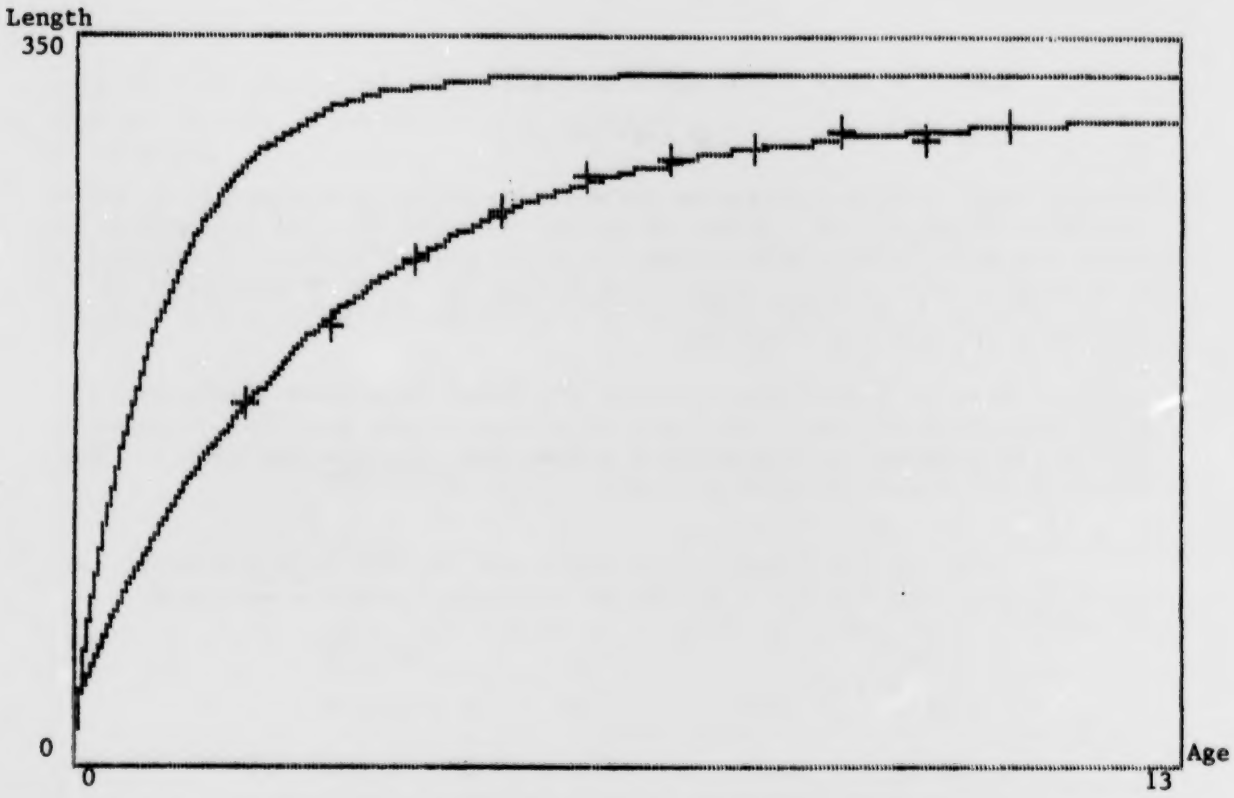


Figure 1.1. Ricker's (1975, p.226) cisco length data (in cm) and two von Bertalanffy growth curves, model (1.1). The data are plotted with + signs. The upper growth curve corresponds to the wild guess $y_\infty = 330$, $K = 1$, and $t_o = 0$. The lower curve corresponds to $y_\infty = 310.8$, $K = .3623$, and $t_o = -.1898$, optimal parameters for the criterion (1.5).

$\hat{y}_\infty$, $\hat{K}$, and $\hat{t}_o$ are determined. A third program, which again accomodates the same user code described earlier, helps this assessment by plotting the criterion function (such as S or S*) as the parameters change. Such a graph is shown in Figure 1.2, where the three curves represent the effect on S due to the following three parameter variations:

$$y_\infty : 310.8 \pm 30 \quad ,$$

$$K : .3623 \pm .1 \quad ,$$

$$t_o : -.1898 \pm .3 \quad .$$

Notice in Figure 1.2 that the changes in y_∞ and K produce similar effects on S, although an increase in K influences S less than a decrease in K. Consequently, a change of ±30 in y_∞ (about ±10%) is comparable to a change of ±.1 in K (about ±25%). It appears that S defines y_∞ better than K. Furthermore, the apparently large change of ±.3 in t_o produces an effect on S which is minor compared to the changes in y_∞ and K. This suggests that S gives weakest definition to the parameter t_o. Incidentally, the program used to produce Figure 1.2 can also be used to study the effect on S of varying two or more parameters simultaneously.

This example illustrates the main features available in the software here. The rest of this report describes exactly how the software works and how to use it. Section 2 outlines the process of searching for a minimum by the simplex method. The user who wishes to understand the algorithm's input and output completely will need to be familiar with the concepts in this section. Section 3 gives the complete list of rules for writing the user program, such as the two lines of code cited above. Section 4 explains how to use MICRO SIMPLEX, the program for minimizing functions like S and S*. Sections 5 and 6 describe PREDICT and PROFILE, the programs which produced Figures 1.1 and 1.2 respectively.

There are many pitfalls in nonlinear estimation and section 7 points out some of them. In particular, this section presents software for implementing ideas by Schnute (1981) to deal with statistical problems associated with a generalized version of the growth model (1.1). Finally, the software here has potential uses which go beyond what has been discussed so far, and section 8 deals with some of them.

Extremely eager readers may wish to see the software at work before reading further. They can, if they wish, jump to Appendix C and follow the instructions for getting the estimates and figures described in this section. It would be best to skim over Appendix A first. As stated in the Preface, this software is designed for use on an Apple II microcomputer. A diskette (called the "system diskette" throughout this report) is available with all the files listed in Appendix B, including the programs shown in full in Appendix D. Write to me, the author, for details on obtaining a copy of the system diskette.

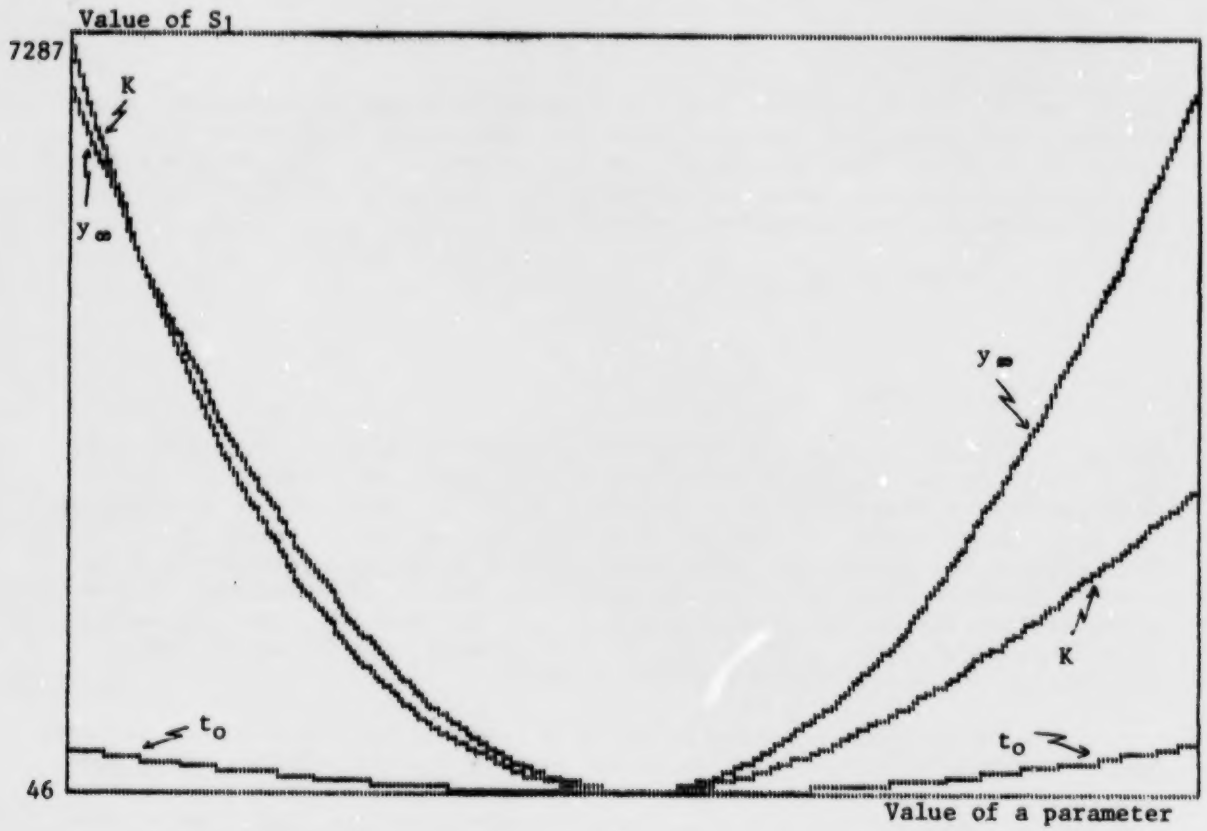


Figure 1.2. Variations in the sum of squares S_1 for Ricker's cisco data as the model parameters deviate from their optimal values. The curve for y_∞ corresponds to values in the range 310.8 ± 30. Similarly the curves for K and t_o correspond to the variations .3623 ± .1 and -.1898 ± .3, respectively.

2. THE SIMPLEX SEARCH METHOD

From the previous section it is clear that a key purpose of the software here is to minimize a function F of several, say N, variables. (In the example above, F is the criterion function S or S*, and N is 3.) There are many known algorithms designed for this purpose, and the most efficient ones require the user to code the derivatives of F, as well as F itself. Other algorithms, called direct search methods, require information on F alone. The simplex method is a direct search method, and the Preface to this report gives some background on my reasons for choosing it. If the practitioner wants to minimize programming time, as opposed to computer run time, then a direct search method is perhaps the best. Users will have to judge from practical operation whether or not this software meets their particular needs.

The simplex method was first conceived by Spendley, Hext, and Himsworth (1962) primarily as a technique for designing experiments to locate points of optimal response for a system being actively measured. Later, Nelder and Mead (1965) pointed out that the method could also be implemented as a computer search. O'Neill (1971) formalized the procedure into a computer FORTRAN algorithm, and the BASIC software here is a variation of O'Neill's algorithm designed for greater flexibility and interactive input/output.

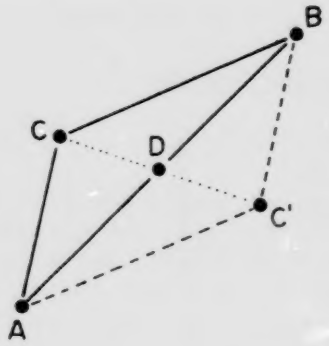
A simplex in N dimensions is polyhedron with N + 1 vertices and N + 1 faces. When N = 2, the simplex is a triangle, which, of course, has 3 vertices and 3 sides. When N = 3, it is a triangular pyramid, that is, a tetrahedron with 4 vertices and 4 triangular faces. The simplex search method involves inspecting the values of the function F at the N + 1 vertices of a simplex. In order to minimize F, the vertex with the highest value of F is rejected in favor of some new point. This new point, together with the remaining N vertices defines a new simplex, and the procedure continues by iteration.

Figure 2.1 illustrates the process when N = 2. Here the triangle ABC represents a two-dimensional simplex on which F is lowest at A and highest at C. Symbolically,

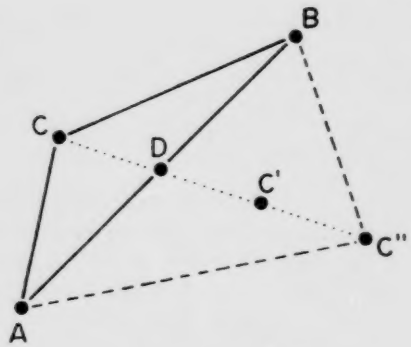
$$F(A) < F(B) < F(C).$$

Since F is highest at C, the idea is to move away from C towards the line AB on which F is lower, at least at the end points. This is done by the process of reflection (Figure 2.1, diagram 1) in which a new point C' is determined as the mirror image of C across the line AB. The point C' is defined so that the lines CC' and AB both have midpoint D.

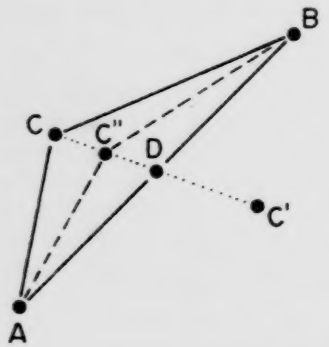
Hopefully, the value of F is lower at C' than at C. Indeed, one can simply count the number of original simplex points A, B, C which have higher F-values than the new point C'. Call this number M. By definition of M, the



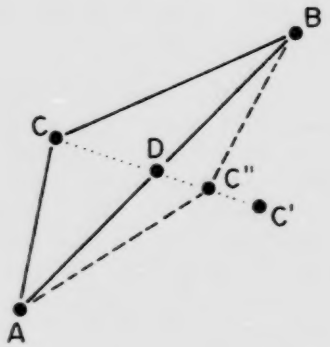
1. REFLECTION



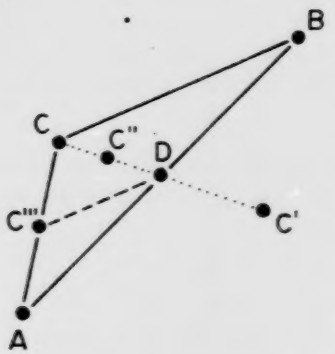
2. EXTENSION



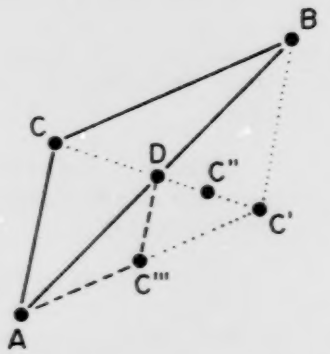
3. CONTRACTION



4. REFLECTION - CONTRACTION



5. REDUCTION



6. REFLECTION - REDUCTION

Figure 2.1. Six possible processes for one iteration of the simplex search method.

function value F(C') at the reflected point C' improves, i.e., is lower than, the value of F at M points on the original simplex ABC. There are four possibilities: M is 0, 1, 2, or 3. The next action taken depends on M.

Case 1: M = 3. In this case the reflected point C' improves all points on the original simplex. Thus the point C' gives the lowest value of F found so far; consequently, the step to C' has been very productive. It is therefore reasonable to extend this step by, say, the distance DC'. This gives C'' in diagram 2 of Figure 2.1. There are two possibilites: either F(C'') is lower than F(C') or not. If so, then the new point C'' is accepted to give a new simplex ABC''. This is called extension. Otherwise, the point C' is accepted to give a new simplex ABC', and, as mentioned above, the process is called reflection (diagram 1).

Case 2: M = 0. If case 1 is the best possible, then this is the worst possible. The reflected point C' is actually worse than any vertex on the original simplex ABC. In this case the choice is made to move from C towards the midpoint D on the line AB, but only half-way. This gives C'' in diagram 3. Hopefully F(C'') is lower than F(C), since a small step has been taken from the highest point C towards the lower points A and B. If so, then C'' is accepted to give the new simplex ABC'' (diagram 3), and the process is called contraction. If, by some weird quirk in the topography, C'' doesn't improve C, then drastic action is required. The decision is then made to shrink the whole simplex towards the lowest point A. Each side of the new simplex ADC''' is half the size of its original counterpart, and the process is called reduction (diagram 5).

Case 3: M = 1. In this case the reflected point C' improves the original point C, but no other. It might seem appropriate simply to accept the reflection (as in diagram 1), but think ahead. The new simplex ABC' would have the property that F is highest at C'; consequently the next iteration would just undo this one to give the old point C as the mirror image of C'. This would be wasted effort, so instead contraction to C'' (diagram 4) is attempted at once. If successful, that is, if C'' improves C', then the process is called reflection - contraction. Otherwise the algorithm performs reflection - reduction to obtain the new simplex ADC''' (diagram 6), analogous to the process of reduction in case 2.

Case 4. M = 2. This is the simplest case. The new point C' is good (since it improves B and C) but not very good (since it doesn't improve A). Consequently, the algorithm proceeds with reflection (diagram 1) in which C' is accepted to give the new simplex ABC'. Incidentally, notice that B is now the high point on the simplex, so the next step will take place in the new direction defined by reflecting B across AC'.

This whole description has been set in two dimensions, but it can be generalized at once to arbitrary N. In the original simplex ABC, let A and C again denote the low and high points; however, let B represent the remaining N-1 vertices at which F takes values between F(A) and F(C). The line AB in Fig. 2.1 now represents a face of the N-dimensional simplex, and D can be regarded as the centroid of this face. The reflected point C' is defined by

reflection of C across D, exactly as before. Again let M denote the number of points in the original simplex which are improved by C'. Then M can take any of the N + 1 values 0, 1, . . ., N. The four cases to consider are:

Case 1. M = N. This case is the best possible, and it might lead to extension.

Case 2. M = 0. This case is the worst possible, and it leads to contraction or even reduction. Incidentally, when N>2, diagram 5 is slightly deceptive, because the centroid D of the face AB differs from the N-1 midpoints of the lines from A to the N-1 vertices B.

Case 3. M = 1. This case is almost as bad, and it leads to reflection - contraction or even reflection - reduction. When N>2, diagram 6 needs interpretation similar to diagram 5 in the reduction step.

Case 3. 1 < M < N. This is the average case, leading to reflection.

The software presented here has as an option the ability to exhibit the value of M, the action taken on each iteration, and the resulting high and low simplex values. Typically, the algorithm moves through cycles of extensions, reflections, contractions, and back to extensions. This occurs because it first finds successful directions, exploits them, and then is forced into a reduced scale before new successful directions can be discovered. The practitioner may find himself or herself drawn into this process, as into a TV show. Watching the algorithm at work is not a bad way to learn something about the model and the data. For example, the parameters best determined become evident during the search. These are the ones which stabilize rather quickly, while the algorithm continues to make larger adjustments to others. Even covariances become somewhat evident. It's quite common to see the algorithm consistently adjust one parameter upward whenever it adjusts another parameter in some particular direction, up or down. The sensitive user may go still further and detect whole groups of parameters whose behavior seems linked during the search for lower F-values.

3. THE USER PROGRAM

To interface with the general software described in this manual, the user must write a program (usually rather short) which describes his or her particular problem. Typically, this involved two functions: (I) the function to be minimized, such as the sum of squares S in (1.4) or S* in (1.8), and (II) the function which defines the model, such as one of the growth models (1.1), (1.2), or (1.3). Each of the three major software packages discussed here requires information about one of these functions. MICRO SIMPLEX, which locates minima, references only the function (I) to be minimized. Similarly

Table 3.1. Special variable names for the user program. The upper section applies to first level rules. Second level rules include the lower section as well.

Label	Significance
F	Function (I) to be minimized; depends on X
X	Parameter vector of dimension N
N	Number of parameters in the function to be minimized
NF	Number of function calculations; also used as a minimization code
YP	Function (II), the value of y predicted by the model; depends on XP and X
XP	Value of independent variable used in prediction
NP	Prediction code
NO	Number of observations*
YO	Vector of observed y-values (dimension NO)*
XO	Vector of observed x-values (dimension NO)*

*Note that the variables NO, YO, and XO involve the letter "O" not the numeral "0" (zero)

PROFILE, which plots function (I) as the parameters vary, references this function only. The program PREDICT is designed to plot observations and model predictions; consequently it references only function (II).

In a typical situation, the function (I) involves function (II) as a sub-computation, but it need not. MICRO SIMPLEX, for example, can be used to minimize any function, not necessarily one based on the statistics of model errors. The user may wish to write a program which interfaces only with MICRO SIMPLEX and PROFILE, but has nothing to do with PREDICT. Such a program would conform to a limited set of rules, referred to here as first level rules. If the program should also interface with PREDICT for plotting function (II), then second level rules also apply.

Perhaps the biggest drawback to BASIC is that the programmer has to be very fussy about variable names. All software here is written with variables ending in the numerals 5, 6, 7, 8, 9, and 0. The user can employ any other names, but some are reserved for special purposes. These are summarized in Table 3.1. Under first level rules, the function F to be minimized is calculated from the parameter vector X of dimension N. The number of function calculations NF is also used as a code for branching to initial and final routines. Under second level rules, the model prediction YP is calculated from the independent variable XP and the parameter vector X. Observations are entered as vectors XO and YO of dimension NO. The prediction code NP is also used for branching to initial steps required by PREDICT.

Figure 3.1 shows the interrelated tasks performed by the user program under each set of rules. First level rules require code to calculate F from X and to perform any initial or final input/output which the user finds necessary or desirable. Second level rules require code to enter any auxiliary model parameters (explained below), to load the observations XO and YO, and to calculate YP from XP and X. This latter calculation may also be regarded as a step in the first level calculation of F from X. Similarly, the entry of auxiliary model parameters and observations may be considered part of first level initialization.

The phrase "auxiliary parameters" refers to parameters specified by the user. These can be contrasted with the N free parameters X(1), X(2), . . ., X(N) to be determined, say, by MICRO SIMPLEX. For example, a linear growth model might be written in the form

$$y = y_1 + a(t - t_1). \qquad (3.1)$$

The line (3.1) passes through the point (t_1,y_1) with slope a. Here t_1 can be regarded as an auxiliary parameter; once t_1 is specified the value of y_1 is determined accordingly. The same line can be represented with any value of t_1; the user simply picks a value which is convenient. For example, the two equations

$$y = 1 + 2(t - 1),$$

$$y = 9 + 2(t - 5),$$

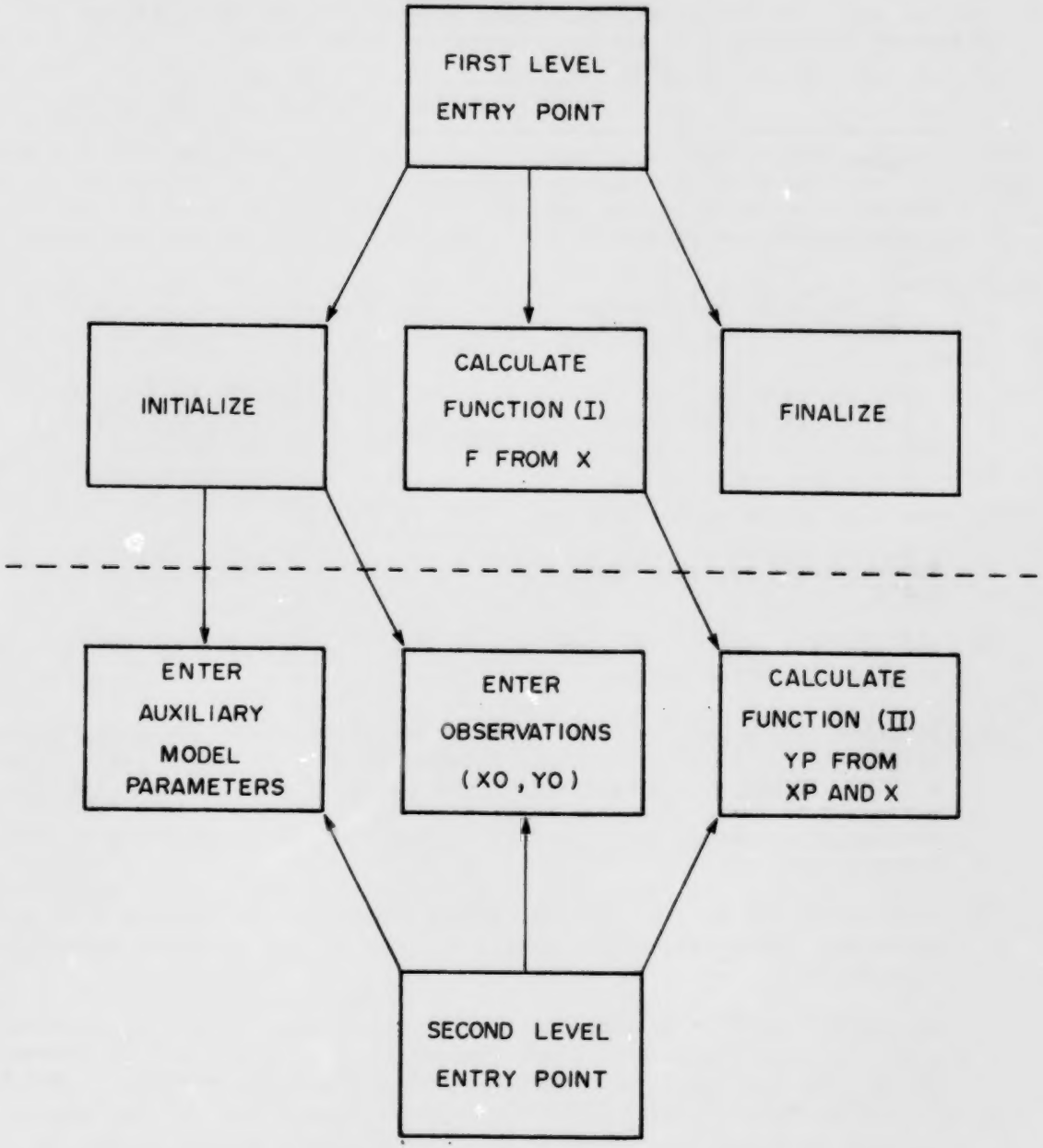


Figure 3.1. Tasks performed by the user program. Those above the dotted line are performed under first level rules. Second level rules involve the tasks below the dotted line as well.

are equivalent. In (3.1), then, there are two free parameters, y_1 and a, and one auxiliary parameter, t_1. Translating (3.1) into the notation discussed here would give the correspondences shown below:

Program	N	X(1)	X(2)	XP	YP
Model	2	y_1	a	t	y

The rules for constructing the user program can now be stated in full. There are seven first level rules and five additional second level rules.

FIRST LEVEL RULES

1. The code should occupy statements 1000 to 39999.

2. Two-character variable names should not end in the numerals 5, 6, 7, 8, 9, and 0.

3. The variable names F, N, X(N), and NF should be reserved for the special meanings described below.

4. The code should have an entry point at statement 1000. Whenever the user program is called at this point, a number NF will be predefined. (During a MICRO SIMPLEX run, NF represents the number of function calculations performed so far). The user code, starting at statement 1000, should consist of a subroutine to perform actions 5, 6, or 7 depending on the value of NF.

5. (Mandatory) If NF > 0, the code should calculate the function F to be minimized. This calculation should be based on the parameter vector X of dimension N.

6. (Optional) If NF = 0, the code should perform any initial input/output which the user requires or finds desirable. This might include messages on the screen, routines to read data and/or auxiliary parameters, and code to choose among possible functions F, such as S or S* in section 1.

7. (Optional) If NF < 0, the code should perform any final input/output which the user finds desirable. This might include a synopsis of the final results, including a reference to the total number (-NF) of function calculations performed during the search. Often this step is unnecessary, because MICRO SIMPLEX itself will produce an optional final summary.

SECOND LEVEL RULES

8. The variable names XP, YP, NO, XO(NO), and NP should be reserved for the special meanings described below.

9. The code should have a second entry point at statement 1500. Whenever the user program is called at this point, a number NP is predefined. The user code, starting at statement 1500, should consist of a subroutine to perform actions 10, 11, or 12 depending on the value of NP.

10. (Mandatory) When NP = 1, the code should calculate the model prediction YP from the independent variable XP and the parameter vector X(N). This process might be part of the calculation in rule 5.

11. (Mandatory) When NP = -2, the code should enter the observed data vectors XO(NO) and YO(NO). These correspond to points (XO(I), YO(I)) for I = 1, 2, . . ., NO, where NO is the number of observations. The code should dimension the vectors XO and YO in the first run only. Typically, data entry is also a part of the initialization described in rule 6.

12. (Optional) When NP = -1, the code should take any actions needed to define the model equation completely. This might include the entry of auxiliary parameters, such as t_1 in (3.1), or the option of selecting among possible models such as (1.1), (1.2), or (1.3). These steps might also be part of the code in rule 6.

As an example, Listing 3.1 shows a reasonably short program which could be used to do all the data analysis for the example cited in section 1. This program conforms to second level rules. Notice in particular the following correspondence between rules and statement numbers:

Rule	Statement
4	1000
5	2000
6	4000
9	1500
10	3000
11	6000

Initialization at 4000 includes data entry at 6000, as well as a choice between additive or logarithmic errors. During the function calculation at 2000, errors are calculated accordingly from the prediction at 3000. No special final output is required, and there are no auxiliary parameters; consequently rules 7 and 12 are not implemented.

As an aid to implementing second level rules, the system diskette includes a file called TEMPLATE. This incorporates all the structure of the program in Listing 3.1 and much more. Often the user needs to append TEMPLATE only with the code (at 3000) for calculating YP. In this particular case,

```
]LIST
1000  REM  ***FIRST LEVEL ENTRY***
1010  IF NF > 0 THEN  GOSUB 2000: REM  CALCULATE F
1020  IF NF = 0 THEN  GOSUB 4000: REM  INITIALIZE
1030  RETURN
1500  REM  ***SECOND LEVEL ENTRY***
1510  IF NP = 1 THEN  GOSUB 3000: REM  PREDICT YP FROM XP
1520  IF NP =  - 2 THEN  GOSUB 6000: REM  GET OBSERVED DATA
1530  RETURN
2000  REM  ***FUNCTION CALCULATION: SUM OF SQUARES***
2010 F = 0
2020  FOR I = 1 TO NO
2030 XP = XO(I): GOSUB 3000
2040  IF E$ = "A" THEN ER = YO(I) - YP
2050  IF E$ = "L" THEN ER =  LOG (YO(I) / YP)
2060 F = F + ER * ER
2070  NEXT I
2080  RETURN
3000  REM ***PREDICT YP FROM XP***
3100 YI = X(1):K = X(2):TZ = X(3):T = XP
3110 YP = YI * (1 -  EXP ( - K * (T - TZ)))
3120  RETURN
4000  REM  ***INITIALIZE***
4010  GOSUB 6000: REM  GET OBSERVATIONS
4020  PRINT : PRINT "ERRORS ADDITIVE OR LOGARITHMIC?"
4030  INPUT "A OR L? ";E$
4040  IF  NOT (E$ = "A" OR E$ = "L") THEN 4030
4050  RETURN
6000  REM  ***READ OBSERVATIONS***
6010  RESTORE :NO = 10
6020  IF RN = 0 THEN  DIM XO(NO),YO(NO):RN = 1
6030  REM  DIMENSION STATEMENT EXECUTED ON FIRST RUN ONLY
6100  DATA  2,172,3,210,4,241,5,265,6,280
6110  DATA 7,289,8,294,9,302,10,299,11,306
6120  FOR I = 1 TO NO
6130  READ XO(I),YO(I)
6140  NEXT I
6150  RETURN
```

Listing 3.1. A user program to do all the data analysis in the example of section 1. This program conforms to second level rules. The optional procedures in rules 7 and 12 are not implemented. Statements 1000, 1500, 2000, 3000, 4000 and 6000 begin the implementation of rules 4, 9, 5, 10, 6, and 11, respectively.

only the two lines

```
3100 YI = X(1) : K = X(2) : TZ = X(3) : T = XP
3110 YP = YI * (1 - EXP (-K * (T - TZ)))
```

are required, as mentioned in section 1. TEMPLATE assumes that the data reside in a file, with a particular structure discussed in Appendix A. Appendix C gives an example which shows precisely how to use TEMPLATE. Typically, the user needs to make only a few modifications, bearing in mind the 12 rules above. It's not a bad idea to inspect all of TEMPLATE when using it, even if only a few lines are added.

As a simple example in this use of first level rules, consider the function

$$G(m,s) = \frac{1}{s^2} \; e^{-\frac{1}{2}\left[\left(\frac{a-m}{s}\right)^2 + \left(\frac{b-m}{s}\right)^2\right]} . \qquad (3.2)$$

This is proportional to the likelihood function for two sample values, a and b, drawn from a normal distribution with mean m and standard deviation s. It's maximum should occur where

$$m = \frac{1}{2}(a + b), \qquad (3.3)$$

$$s = \sqrt{\frac{1}{2}\left[(a-m)^2 + (b-m)^2\right]} = \frac{|a-b|}{2} . \qquad (3.4)$$

Listing 3.2 shows a suitable program to use MICRO SIMPLEX for approximating m and s by maximizing the likelihood G in (3.2). A final output routine (rule 7) compares the estimates to the correct values given by (3.3) - (3.4). Notice that G is maximized by minimizing the function F = -G.

The actual use of a program, such as Listing 3.1 or 3.2, requires merging it with one of the system software packages, such as MICRO SIMLPEX, PROFILE, or PREDICT. Merging can be accomplished in several ways, discussed in Appendix A, item 2.

For the purpose of debugging a user program, the system diskette includes the file TRY. It will put a user program through the paces implied by first or second level rules. Listings 3.3 and 3.4 show two runs of TRY when it has been merged with the user program for von Bertalanffy estimation shown in Listing 3.1. At the outset, TRY wants the user to choose between first level entry (function calculation F) or second level entry (prediction P). As in Listing 3.3, the "F" option should lead to output confirming that initialization (rule 6), function calculation (rule 5), and final output (rule 7) have performed successfully. Similarly, as in Listing 3.4, the "P" option should verify the processes of data entry (rule 11), auxiliary parameter entry

```
]
]LIST

1000  REM  ***FIRST LEVEL ENTRY***
1010  IF NF = 0 THEN 2000
1020  IF NF > 0 THEN 3000
1030  IF NF < 0 THEN 4000
2000  REM  ***INITIAL INPUT***
2010  INPUT "WHAT IS A? ";A
2020  INPUT "WHAT IS B? ";B
2030  RETURN
3000  REM  ***CALCULATE F = -G***
3010 M = X(1):S = X(2)
3020 G = ((A - M) / S) ↑ 2 + ((B - M) / S) ↑ 2
3030 G =  EXP ( - G / 2) / (S * S)
3040 F =  - G
3050  RETURN
4000  REM  ***FINAL OUTPUT***
4010  PRINT : PRINT "APPROXIMATE MAX. LIKELIHOOD ESTIMATES:"
4020  PRINT "MEAN M",X(1)
4030  PRINT "STD. DEV. S",X(2)
4040  PRINT "TRUE MAX. LIKELIHOOD ESTIMATES:"
4050  PRINT "MEAN M",(A + B) / 2
4060  PRINT "STD. DEV. S", ABS (A - B) / 2
4070  PRINT : RETURN
```

Listing 3.2. A user program to find maximum likelihood estimates for the mean M and standard deviation S from a sample of two values, A and B. This program conforms to first level rules. Statements 1000, 2000, 3000, and 4000 begin the implementation of rules 4, 6, 5, and 7, respectively.

```
]RUN
PROGRAM TO TEST USER ROUTINE
  FOR MICRO SIMPLEX

TEST FOR PREDICTION OR
  FUNCTION CALCULATION?
P OR F? F                                        User response

WHAT IS THE NUMBER OF PARAMETERS
  OR COORDINTES?
N = 3                                            User response

ENTER A TRIAL PARAMETER VECTOR:
?330                                             User response
?1                                               User response
?0                                               User response

ERRORS ADDITIVE OR LOGARITHMIC?                  User program
A OR L? A                                        User response to user program

*** USER'S INITIALIZATION COMPLETED.

*** FUNCTION EVALUATION COMPLETED.
POINT:
330    1    0
FUNCTION VALUE: 42076.0222

PRESS ANY KEY TO CONTINUE . . .

*** USER'S FINAL OUTPUT COMPLETED.

ENTER A TRIAL PARAMETER VECTOR:
?311                                             User response
?.362                                            User response
?-.2                                             User response

*** FUNCTION EVALUATION COMPLETED.
POINT:
311    .362    -.2
FUNCTION VALUE: 47.1643433

PRESS ANY KEY TO CONTINUE . . .

*** USER'S FINAL OUTPUT COMPLETED.

ENTER A TRIAL PARAMETER VECTOR:
```

Listing 3.3. TRYing the program in Listing 3.1. Selecting the function calculation (F) option exercises first level rules. The right-hand column shows the source of control for each line not printed entirely by TRY.

```
]RUN
PROGRAM TO TEST USER ROUTINE
  FOR MICRO SIMPLEX

TEST FOR PREDICTION OR
  FUNCTION CALCULATION?
P OR F? P                                    User response

*** OBSERVED DATA ENTERED.
10 OBSERVATIONS
FIRST:          2,172
LAST:           11,306

LIST THEM ALL?
Y OR N? Y                                    User response
OBS X           OBS Y
2               172
3               210
4               241
5               265
6               280
7               289
8               294
9               302
10              299
11              306

*** AUXILIARY MODEL PARAMETERS ENTERED.

WHAT IS THE NUMBER OF PARAMETERS
  OR COORDINTES?
N = 3                                        User response

ENTER A TRIAL PARAMETER VECTOR:
?311                                         User response
?.362
?-.2

ENTER A VALUE OF X FOR PREDICTION:
XP = 5                                       User response

*** PREDICTION COMPLETE.

AT X = 5, Y = 263.658234

ENTER A TRIAL PARAMETER VECTOR:
```

Listing 3.4. TRYing the program in Listing 3.1. Selecting the prediction (P) option exercises second level rules. The right-hand column shows the source of control for each line not printed entirely by TRY.

(rule 12), and prediction (rule 10). Notice that output from TRY gives the user a chance to check that his or her program does indeed calculate F and YP correctly from the parameter vector X and the independent variable XP. A thorough check would involve a completely separate hand calculation.

4. MICRO SIMPLEX

This program is designed to wed the simplex search method described in section 2 with a function coded according to first level rules in section 3. The purpose, once again, is to locate an N-dimensional vector X at which the function F(X) is minimized. Like any nonlinear search method, the simplex method requires tuning to the particular problem. For this reason MICRO SIMPLEX includes two phases: (1) entry and adjustment of the tuning parameters, and (2) the simplex search itself. Sometimes the search process indicates a need for re-tuning; consequently, MICRO SIMPLEX allows easy movement between phases (1) and (2).

The key to a successful application of MICRO SIMPLEX lies in proper tuning. This requires a thorough understanding of the significance of the input parameters described in the next few paragraphs. Listing 4.1 is included here for illustration. It shows the process of entering data necessary for estimating von Bertalanffy parameters by using MICRO SIMPLEX merged with the program in Listing 3.1.

1. Initial point and step.

The simplex algorithm described in section 2 needs a simplex with which to begin. The initial point and step specify this simplex. For example, in Listing 4.1 there are three variables corresponding to the von Bertalanffy parameters y_∞, K, and t_0. The initial point, a wild guess with $y_\infty = 400$, $K = 1$, and $t_0 = 0$, specifies one vertex of the required 3-dimensional simplex. The other three vertices (for a total of four) are obtained from the initial step (50, .2, .2). In this case these remaining vertices are (450, 1, 0), (400, 1.2, 0), and (400, 1, .2). A guideline for selecting the initial step is to pick step sizes in the variables which reflect an intuitive sense of their uncertainty. When $y_\infty = 400$ cm, then 50 cm represents a fairly large change. The parameter K is an exponential growth parameter, and a change of .2 in the estimate 1 seems reasonable. Finally t_0, which represents an age in years, might be something like ± 2, ± 1, or 0. Here 0 is an initial estimate with a step of .2. A larger step would also be quite reasonable. There is no reason to be terribly concerned over the choice of initial point and step. Usually the search quickly adjusts the simplex as required. If the initial step is small, the process typically begins with extensions. Similarly, a large initial step may cause the search to begin with contractions.

```
JRUN
              MICRO SIMPLEX

A COMPLETE PROGRAM TO LOCATE
MINIMA FOR ARBITRARY FUNCTIONS.
SEE SEPARATE DOCUMENTATION.


  -- BY JON SCHNUTE.

NAME OF ALGORITHM DATA FILE?                  This file will save the data
?A.DATA                                         entered here to allow easy
DATA ALREADY ON FILE?                           re-entry in future runs.
Y OR N? N
ENTER THE ALGORITHM DATA NOW.

NUMBER OF VARIABLES?                          Data entry is prompted by
?3                                              MICRO SIMPLEX.
INITIAL POINT?
?400
?1
?0
INITIAL STEP?
?50
?.2
?.2
REQUIRED S.D. FOR FUNCTION VALUES?
?.00001
MAX. NO. OF FUNCTION CALLS?
?500
ITERATIONS BET. CONSOLE DISPLAYS?
(0 SUPPRESSES DISPLAY.)
?1
ITERATIONS BET. PRINTER DISPLAYS?
(0 SUPPRESSES PRINTER COMPLETELY.)
?20
ITERATIONS BET. CONVERGENCE CHECKS?
?1
STEP REDUCTION FRACTION FOR RESTART?
?.001

3 VARIABLE(S)                                 Automatic summary of data
INITIAL POINT:                                  entered.
400    1    0
INITIAL STEP:
50    .2    .2
REQUIRED STD. DEV. = 1E-05
MAX. OF 500 FUNCTION CALLS
CONSOLE DISPLAY CODE = 1
PRINTER DISPLAY CODE = 20
CONV. CHK. EACH 1 ITERATIONS
STEP REDUCED BY 1E-03 ON RESTART

EVERYTHING OK?
```

Listing 4.1. (Two pages. See caption next page.)

```
Y OR N? N
RE-ENTER SOME OR ALL DATA?
S OR A? S
ENTER CODE FOR CHANGED DATA.
CD - GIVES LIST OF CODES.
NO - NO MORE CHANGES.
?CD
IP - INITIAL POINT                                  Two-letter codes for
IS - INITIAL STEP                                     selecting individual
SD - REQ. S.D. FOR I.N. CALLS                         data items to be changed.
MF - MAX. NO. OF FUNCTION CALLS
CO - CONSOLE DISPLAY CODE
PR - PRINTER DISPLAY CODE
CC - ITERS. BET. CONVERGENCE CHECKS
SR - STEP REDUCTION ON RESTART
CD - LIST OF CODES
NO - NO MORE CHANGES
ENTER CODE FOR CHANGED DATA.
CD - GIVES LIST OF CODES.
NO - NO MORE CHANGES.
?IQ                                                 User enters inappropriate
ENTER CODE FOR CHANGED DATA.                          code and is prompted to
CD - GIVES LIST OF CODES.                             try again.
NO - NO MORE CHANGES.
?IP                                                 Code to re-enter initial
NEW INITIAL POINT:                                    point.
?330
?1
?0
ENTER CODE FOR CHANGED DATA.
CD - GIVES LIST OF CODES.
NO - NO MORE CHANGES.
?NO

3 VARIABLE(S)                                       Automatic summary of
INITIAL POINT:                                        amended data.
330    1    0
INITIAL STEP:
50    .2    .2
REQUIRED STD. DEV. = 1E-05
MAX. OF 500 FUNCTION CALLS
CONSOLE DISPLAY CODE = 1
PRINTER DISPLAY CODE = 20
CONV. CHK. EACH 1 ITERATIONS
STEP REDUCED BY 1E-03 ON RESTART

EVERYTHING OK?
```

Listing 4.1. (Second page) Sample input for MICRO SIMPLEX merged with the von Bertalanffy estimation program in Listing 8.1. The first page illustrates data entry. This second page illustrates how to change some of the data entered. The right-hand column on each pages adds commentary to the listing.

2. Required standard deviation for function values on the simplex.

The search needs a criterion for stopping. Intuitively, it should stop when it can't manage to reduce the function F much further. When this happens, the value of F is about the same on each of the N+1 vertices of the simplex. The standard deviation of these N+1 F-values measures the amount of variation. Consequently, it's reasonable to terminate the search when this quantity becomes small enough. The user must specify what "small enough" means. In Listing 4.1, it is taken to be 10^{-5}. An appropriate choice for the standard deviation depends on the value of F itself. In the example cited here, the minimum F turns out to be about 46, so that 10^{-6} measures the eighth significant figure. Unfortunately, the user doesn't always know approximate values of F at first (although TRYing the user code gives some idea). In such cases it may be necessary to do a short run and then reset the required standard deviation.

3. Step reduction fraction for restart.

Once a sufficiently small standard deviation in F-values is reached, the algorithm performs a minimum check. This involves changing each of the final (hopefully optimal) N parameters by plus and minus a fixed amount. The amount is equal to the initial step size times some fixed fraction. In the example here, the step is (50, .2, .2) and the fraction is .001. This means that the final parameters will be varied by the amounts (±.05, ±.0002, ±.0002). Table 4.1 shows the outcome of the test in this case. The value of F is slightly higher at each of the six points tested than at the minimum found. Consequently, the minimum is accepted, and the algorithm stops. If a lower F-value were found at one of the six points tested, then the process would restart using this as the initial point and the reduced step (.05, .0002, .0002) as the initial step.

4. Maximum number of function calls.

This number simply sets an upper limit to the number of function calculations allowed. When the limit is reached, the algorithm stops even if no minimum is found.

5. Iterations between console displays, or console display code.

On request, MICRO SIMPLEX will automatically put information on the screen about algorithm performance. The information displayed depends on the value of the number entered here. A console display code 0 suppresses any listing. If the code is 1, then after every iteration the screen will show the current low point, the current simplex high and low F-values, and the action (reflection, extension, contraction, etc.) just taken. If the code is an integer m, then the same information, except for the action just taken, will be displayed every m iterations. Whenever the code is not 0, there will also be initial and final displays of pertinent information. In particular, the final display states the reason why the algorithm terminated.

6. Iterations betwen printer displays, or printer display code.

This has the same effect on the printer as the previous code does on the console. It is assumed that the printer is located in Apple slot no. 1. If not, simply change line 170 of MICRO SIMPLEX (version 1.1) accordingly. If no printer is available, always set this code to 0. As in listing 4.1, it is

Table 4.1 Minimum test for the sum of squares S_1 and Ricker's (1975) cisco length data. The first entry represents the minimum found. (See section 1 and Listing 4.3). The remaining six points represent parameter variations $\pm.05$, $\pm.0002$, $\pm.0002$ in y_∞, K, and t_o, respectively.

y_∞	K	t_o	$F = S_1$
310.822	.36231	-.18978	46.36998
310.772	.36231	-.18978	46.38818
310.872	.36231	-.18978	46.38928
310.822	.36211	-.18978	46.38753
310.822	.36251	-.18978	46.38775
310.822	.36231	-.18958	46.37020
310.822	.36231	-.18998	46.37017

often useful to set the console display code to 1 for a continual view of the details, and to set a higher print code (say 20) to keep a record of general algorithm performance.

7. Iterations between convergence checks.

Minor amounts of computer time can be saved by not bothering to compute the standard deviation in F-values each iteration. Setting this code to m results in a convergence check only every m iterations. In most cases, the great majority of time is spent calculating F in the user program. Consequently, there is usually little advantage in choosing m > 1.

When MICRO SIMPLEX is merged with the user program and run, it begins at once with the data entry phase just described. As Listing 4.1 shows, the first step requires the user to specify a file for saving the algorithm data. The example here uses the name A.DATA. This file is to contain all the data items 1 to 7 above, and it is generated and read entirely by MICRO SIMPLEX. It should be clearly distinguished from any file referenced by the user program, such as a file of observed data (XO(I),YO(I)), I = 1, . . ., NO. Therefore, a warning: do not enter the name of an observed data file at this point, or you may destroy its contents.

MICRO SIMPLEX asks the user if the algorithm data is already on file from some previous run. If so, it is entered automatically. If not, then the user is prompted to enter the data, item by item. Once this is done, the screen shows a summary list. Should the user notice any necessary changes, he or she can specify a two-letter code designating the item to be changed. The code "CD" gives a list of codes. Listing 4.1 shows the use of the code "IP" to change the initial point from (400, 1, 0) to (330, 1, 0).

When the algorithm data is finally entered correctly, the user indicates that everything is OK, and MICRO SIMPLEX executes the user's own initial code (rule 6 in section 3). Then computation begins. Listing 4.2 shows the results of the first four iterations in the von Bertalanffy example (Listing 3.1), assuming algorithm data as in Listing 4.1 and additive (as opposed to logarithmic) errors. The reader who understands the simplex method, particularly Figure 2.1, will be able to account for the number of function calls in each iteration. The performance here is quite typical. Successful steps in the first few iterations result in an extended scale. Then, in the fourth iteration here, a reduced scale is necessary. Listing 4.3 shows printer output from the complete run. The final answer is the one cited in section 1.

When MICRO SIMPLEX finishes, it offers the option of starting again, using the current minimum as the starting point for a new run. This can be particularly convenient when the algorithm stops because the maximum number of function evaluations is reached, or when convergence fails because of improper tuning. Furthermore, algorithm performance can be modified mid-stream. Simply interrupt the program by pressing CTRL-C, and then modify one of the following parameters:

```
INITIAL POINT:
330    1    0
INITIAL FUNCTION:  42076.0222
INITIAL SIMPLEX HI AND LO:
120932.255    38884.6131

6 CALLS, 1 ITERS., 0 RESTARTS                    4 calls for building the
VALUE AT REFLECTED POINT IMPROVES                  initial simplex plus two
  4 VALUE(S) AT SIMPLEX POINTS                     calls for reflection and
ACTION(S) ACCEPTED THIS ITERATION:                 attempted (but failed)
  REFLECTION                                       extension.
CURRENT MINIMUM FOUND AT:
280    1.13333333    .133333333
CURRENT HI AND LO:
48046.9099    12263.101

8 CALLS, 2 ITERS., 0 RESTARTS                    2 calls for reflection and
VALUE AT REFLECTED POINT IMPROVES                  extension.
  4 VALUE(S) AT SIMPLEX POINTS
ACTION(S) ACCEPTED THIS ITERATION:
  EXTENSION
CURRENT MINIMUM FOUND AT:
280    .733333333    .333333333
CURRENT HI AND LO:
42076.0222    3933.105

9 CALLS, 3 ITERS., 0 RESTARTS                    1 call for reflection.
VALUE AT REFLECTED POINT IMPROVES
  3 VALUE(S) AT SIMPLEX POINTS
ACTION(S) ACCEPTED THIS ITERATION:
  REFLECTION
CURRENT MINIMUM FOUND AT:
280    .733333333    .333333333
CURRENT HI AND LO:
38884.6131    3933.105

11 CALLS, 4 ITERS., 0 RESTARTS                   2 calls for reflection and
VALUE AT REFLECTED POINT IMPROVES                  contraction.
  0 VALUE(S) AT SIMPLEX POINTS
ACTION(S) ACCEPTED THIS ITERATION:
  CONTRACTION
CURRENT MINIMUM FOUND AT:
280    .733333333    .333333333
CURRENT HI AND LO:
15171.1422    3933.105
```

Listing 4.2. The first four iterations in the search for von Bertalanffy parameters for Ricker's cisco length data. Here MICRO SIMPLEX has been merged with the program in Listing 3.1, and the data in Listing 4.1 have been entered. The right-hand column gives explanation of the numbers of function calls.

```
3 VARIABLE(S)
INITIAL POINT:
330    1    0
INITIAL STEP:
50    .2    .2
REQUIRED STD. DEV. = 1E-05
MAX. OF 500 FUNCTION CALLS
CONSOLE DISPLAY CODE = 1
PRINTER DISPLAY CODE = 20
CONV. CHK. EACH 1 ITERATIONS
STEP REDUCED BY 1E-03 ON RESTART

INITIAL POINT:
330    1    0
INITIAL FUNCTION:  42076.0222
INITIAL SIMPLEX HI AND LO:
120932.255    38884.6131

40 CALLS, 20 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
302.702529    .476943889    .40242806
CURRENT HI AND LO:
342.973333    255.53261

76 CALLS, 40 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
313.406405    .370183374    .036777062
CURRENT HI AND LO:
170.560804    107.208872

112 CALLS, 60 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
310.693802    .363027668    .186921011
CURRENT HI AND LO:
46.4876447    46.3924423

150 CALLS, 80 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
310.82173    .362311926    .189776266
CURRENT HI AND LO:
46.3700089    46.3699821

158 CALLS, 82 ITERS., 0 RESTARTS
FINAL POINT PASSED MINIMUM TEST
MINIMUM FOUND AT:
310.82173    .362311926    .189776266
FN. VALUE AT THIS POINT:  46.3699821
```

Listing 4.3. Printer output from a complete run of MICRO SIMPLEX, merged with the program in Listing 3.1. The algorithm data are printed first. The printer display code 20 gives output each 20 iterations, as well as initial and final results.

M0 maximum number of function calls
C0 console display code
P0 printer display code
R5 required standard deviation
S5 step reduction fraction

For example, setting M0 = 1 will terminate the search in an orderly way and bring the user back to the data entry phase. Similarly, a change in C0 or P0 leads to an increase or decrease in the output during part of the search. Finally, the user can actively tune the search by adjusting R5 and S5.

The values of R5 and S5 are the key to proper tuning. The required standard deviation R5 sets limiting degree of precision in the function F. Similarly, the step reduction fraction, together with the initial step, specifies the precision in each parameter. These two quantities are related in the sense that a small change in one parameter produces a corresponding change in the function F. Table 4.1 illustrates this; notice that a change in the fourth or fifth signifcant digit of y_∞, K, and t_0 produces a change in the fifth or sixth signifcant digit in F. Consequently, the required standard deviation of 10^{-6} (the eighth significant digit of F) is probably more than adequate to ensure a precision of y_∞, K, and t_0 to within the fraction 10^{-3} of the initial step (50, .2, .2).

The essential point to recognize here is that MICRO SIMPLEX first checks for convergence by estimating the precision of F from its standard deviation on the current simplex. If the desired precision is reached, then a minimum check tests whether the final parameter vector X does indeed appear to be a minimum to the precision defined by the initial step and the step reduction. This second check is costly because it involves up to 2N function calculations, as in Table 4.1. The idea, then, is to pick a required standard deviation small enough so that the final minimum check will succeed. On the other hand, too small a required standard deviation will force the search to continue long after the parameter vector X has been determined to a reasonable precision.

Usually a certain amount of experience is required to set R5 and S5 properly in a particular problem. If the search continues for many iterations after the parameter vector X becomes relatively stable, then the required standard deviation R5 is probably set too small. On the other hand, if the algorithm automatically restarts because the minimum test fails, then R5 should be smaller. Even though automatic restart is built-in, the user should regard restarts as warnings of improper tuning.

Ultimately, the search is limited by the available precision in the computer. In some cases the user may specify an extremely small required standard deviation R5 in the attempt to obtain high precision in the parameter vector X. When the calculation of F begins to be limited by computer precision, the function no longer appears continuous. For example, when F no longer varies except in the last available digit of precision, it can

effectively take only ten values (rather than a continuum of values). The search algorithm sees very peculiar function behavior and is forced to do many reductions, as described in section 2. Therefore, frequent reductions are usually a clue that computer precision has become limiting, and it may not be possible to get better precision in the parameter vector X because changes in F cannot be detected. Applesoft is a comparatively useful language in this regard, because its five-byte variables have nine or ten significant figures. Many other BASIC implementations use four-byte variables with only about six significant figures.

In summary, here is a short guide to proper tuning:

1. If the search continues long after the parameter vector X stabilizes to a degree satisfactory to the user, the required standard deviation is probably set too small.

2. If the search resorts to many restarts (which are slow and inefficient), the required standard deviation is probably set too large.

3. If the search leads to many reductions (which are also slow and inefficient), computer precision has probably become limiting. In this case either condition 1 above is the problem, or else the user needs to re-assess the precision needed in the parameter vector X. The function F itself may be a poor measure of information, and in some cases the statistical model may be inappropriate.

4. When the algorithm is properly tuned, it should locate a minimum without restarting, and it should use primarily reflections, extensions, and contractions.

MICRO SIMPLEX contains a subroutine in lines 50000 to 59999 which performs the actual search and displays details of the process on request. The user can incorporate this subroutine into an extension of the user program described in section 3. This extended program would not only evaluate the function F, but also set up the input parameters to the subroutine. These are listed in Table 4.2. Such a program might be used, for example, to do many successive parameter estimations without interruption. Typically the user might first take advantage of the full program MICRO SIMPLEX to locate the proper parameters for tuning. Then a revised user program would automatically set the correct tuning parameters and call the search algorithm subroutine to find a minimum. Typically, the display codes CO and PO might be set to 0 in order to suppress output during production runs.

5. PROFILE

This program plots F as a function of X when F(X) is defined by user

Table 4.2. Input/output to the simplex search subroutine within MICRO SIMPLEX. This subroutine occupies lines 50000 to 59999.

ALGORITHM INPUT DATA	
N	Number of coordiantes or parameters
XO(N)	Initial point
SO(N)	Initial step
R5	Required standard deviation for function values
S5	Step reduction fraction or restart
MO	Maximum number of function calls
CO	Console display code
PO	Printer display code
I5	Iterations between convergence checks
ALGORITHM OUTPUT DATA	
F	Minimum function value found
X(N)	Point at which minimum F was found
NF	Negative of the total number of function calls
IO	Total number of iterations
RO	Total number of restarts
E5	Exit code (-1 if too many function calls, or 1 if minimum test passed)

code which conforms to first level rules (section 3). Because X is a vector of dimension N, where N might be as large as 10 or 15, it is impossible to give two-dimensional representation to the full graph of F(X). Instead, the program focuses on the behavior of F along a line in X-space. This behavior can be represented by an ordinary two-dimensional curve with the vertical coordinate corresponding to the value of F and the horizontal coordinate corresponding to the value of X on some particular line in N-dimensional space. It's reasonable to refer to such a curve as a "profile" of F. This concept gives the program its name.

Typically the user wants to examine the behavior of F near a minimum point X determined by MICRO SIMPLEX. This point might be regarded as the mid-point of a line segment prescribed also by the step to the left- and right-hand end points. For instance, in the example of section 1, the point

X = (310.8, .3623, -.1898)

gives optimal estimates for y_∞, K, and t_0. To study the effect of altering y_∞ by ±30, the user might consider the step (30, 0, 0). This defines a line segment with end points (280.8, .3623, -.1898) and (340.8, .3623, -.1898).

Figure 1.2 (page 6), discussed earlier in the introductory section, shows this particular profile for the sum of squared additive errors S_1. Two other profiles are also shown there, representing changes of ±.1 and ±.3 in K and t_0, respectively. Listing 5.1 shows the process of entering the data to produce Figure 1.2, under direction of the program PROFILE merged with the user program in Listing 3.1 (page 16). All prompts for data come from PROFILE, except the line in which additive, rather than logarithmic, errors are chosen. This option comes from the initializing routine in the user program, which PROFILE automatically executes.

As Listing 5.1 shows, PROFILE allows the user to select how many profiles (here 3) will be produced during the run. All have a common mid-point, but the step to the end points is different for each. The program prompts to make sure that each vector is entered correctly. If not, type "N". The reply "Y", or even a null entry, indicates that the data just entered are OK. Profiles usually take quite a while to calculate. For example, here the 3 profiles with 60 points per profile, require 180 calculations of S_1. This actually exceeds the number of function calls (158) used by MICRO SIMPLEX to locate the minimum point. (See Listing 4.3.)

The examples cited so far all involve steps in which only one variable (here y_∞, K, or t_0) is changed. Such steps are perhaps of the most common interest, although PROFILE is not restricted to them. It might also be interesting, for example, to study how S_1 changes when both y_∞ and K are changed, with t_0 left fixed. Two possibly interesting steps might be (30, .1, 0) and (30, -.1, 0). Examining these steps would indicate whether S_1 is influenced more by changes of y_∞ and K in the same or opposite directions.

When PROFILE finally finishes its work, the computer signals with a beep. The user can, on request, produce a printed description of the

```
]RUN
PROGRAM TO PLOT 'PROFILES' OF A
USER-DEFINED FUNCTION OF N VARIABLES.
THE FUNCTION SHOULD BE CODED AS
REQUIRED FOR MICRO SIMPLEX.

A PROFILE IS A GRAPH OF FUNCTION
VALUES ALONG A LINE BETWEEN TWO
SPECIFIED POINTS IN N-SPACE.

WARNING!...THIS PROGRAM SHOULD BE                Very important!
LOADED WITH HI-RES GRAPHICS PAGE 1
PROTECTED. USE PROGRAM 'PROTECT'
TO ACHIEVE THIS.

WHAT IS N? 3
HOW MANY PROFILES IN THIS RUN? 3
HOW MANY POINTS ON EACH PROFILE?
(MIN 2, MAX 280, SUGGEST 20 OR MORE)
?60
ENTER COORDINTES OF THE MID-POINT
  FOR ALL PROFILES:
?310.8
?.3623
?-.1898
HERE ARE THE COORDINATES JUST ENTERED:
310.8    .3623    -.1898
OK? (Y OR N) Y                                   Null entry taken to mean "Y".
ENTER THE COORDINATES OF THE STEP TO
  THE FINAL POINT FOR PROFILE 1:
?30
?0
?0
HERE ARE THE COORDINATES JUST ENTERED:
30   0   0
OK? (Y OR N)
ENTER THE COORDINATES OF THE STEP TO
  THE FINAL POINT FOR PROFILE 2:
?0
?.1
?0
HERE ARE THE COORDINATES JUST ENTERED:
0   .1   0
OK? (Y OR N)
ENTER THE COORDINATES OF THE STEP TO
  THE FINAL POINT FOR PROFILE 3:
?0
?0
?.3
HERE ARE THE COORDINATES JUST ENTERED:
0   0   .3
```

Listing 5.1. **(Two pages. See caption next page).**

```
OK? (Y OR N)

ERRORS ADDITIVE OR LOGARITHMIC?
A OR L? A

PROFILE CALCULATIONS BEGIN. THIS
MAY TAKE A WHILE...


PROFILE CALCULATIONS COMPLETE.
DO YOU WANT A SUMMARY OF PROFILE
  DATA ON THE PRINTER?
Y OR N? Y
PREPARE PRINTER, 'RETURN' TO START...


MID-POINT FOR ALL PROFILES:
310.8    .3623    -.1898

PROFILE 1 BEGINS AT:
280.8    .3623    -.1898
AND ENDS AT:
340.8    .3623    -.1898

PROFILE 2 BEGINS AT:
310.8    .2623    -.1898
AND ENDS AT:
310.8    .4623    -.1898

PROFILE 3 BEGINS AT:
310.8    .3623    -.4898
AND ENDS AT:
310.8    .3623    .1102

PROFILE          MINIMUM          MAXIMUM
#1               48.1377993       6806.33258
#2               47.5096346       7287.26146
#3               46.4734231       553.322923
ALL              46.4734231       7287.26146
```

This option comes from the initializing routine in the user program.

Listing 5.1. (Second page) Input to PROFILE merged with the program in Listing 3.1, along with a description of the resulting graph. This set-up gives profiles which represent the behavior of the sum of squared additive errors S_1 for Ricker's cisco length data and model (1.1) when y_∞, K, and t_o are varied by ±30, ±.1, and ±.3, respectively, from the estimates 310.8, .3623, -.1898. The graph of the three profiles described here is shown earlier in Figure 1.2.

profiles, as shown in Listing 5.1. The sensitive reader may notice that in this case the minimum is not the same on all three profiles. Theoretically, it should be, since all profiles share a common midpoint which is an overall minimum for F. In fact, F is calculated at only 60 points on each profile, and the exact midpoint may not be one of them.

After the printer summary is completed (if requested), PROFILE moves to a new mode which allows graphical display. The user sees a blank graphics page, with border only, and a series of prompts: P)LOT, E)RASE, S)AVE, L)OAD, or Q)UIT. Typing one of the letters P, E, S, L, or Q determines the next action. Any other keystroke simply reproduces the prompt.

The most common choice is P)LOT. The user is asked which variable to plot. In the example here this means a number between 1 and 3. Typing "1" gives the first profile, here the one resulting from a change ± 30 in y_m. Similarly profiles #2 and #3 correspond to changes in K and t_o respectively. The user actually sees the profiles traced across the screen, so it is possible to distinguish each one from the others. Furthermore, the E)RASE option allows the screen to be cleared so that plotting can begin again. This means that profiles can be viewed singly, or in arbitrary pairs, triplets, etc.

If the user has a suitable printer, he or she may wish to produce hard copy of some picture, like Figure 1.2. Unfortunately each model printer has its own special software to reproduce the graphics page. The philosophy in PROFILE is to S)AVE the current picture in a diskette file for later printing by the user's particular software. The file name specified by the user in response to S)AVE will be prefixed "PRO.". For example, the name "FIRST PICTURE" will result in a diskette file "PRO. FIRST PICTURE" containing the picture currently on the screen (graphics page 1). Similarly, if the user L)OADs a previous picture, the prefix "PRO." is automatically supplied. A minor warning: before using the S)AVE option, be sure the active drive has enough space available to hold a graphics page.

One final very important note on the use of PROFILE: don't load and run it until graphics page 1 is protected from containing parts of the program. Ordinarily, APPLESOFT uses memory locations below this graphics page and the program can (and will in the case of PROFILE) overlap the graphics with catastrophic results. The unfortunate user has patiently entered all necessary data and waited until profiles are calculated, only to watch the system crash as soon as the plotting phase is reached. The system diskette contains the program PROTECT to reserve graphics pages. Always RUN PROTECT and reserve page 1 before using PROFILE. (See also the warning note at the start of each run, as in Listing 5.1). In some cases, it may also be necessary to use PROTECT to free page 1 before another MICRO SIMPLEX run, because MICRO SIMPLEX is larger, needs more space, and doesn't use graphics.

```
DO YOU WANT A PRINTED DATA SUMMARY?
Y OR N? N
SAVE THIS PICTURE ON DISK?
Y OR N? N
PLOT ANOTHER PREDICTION WITH
  DIFFERENT PARAMETERS?
Y OR N? Y

ENTER THE PARAMETERS:
?330
?1
?0

HERE ARE THE VALUES ENTERED:
330    1    0
OK? (Y OR N) ?Y


X: 0 TO 13
Y: 0 TO 350
PRESS ANY KEY TO CONTINUE...
DO YOU WANT A PRINTED DATA SUMMARY?
Y OR N? Y

OBSERVATIONS:
     X FROM 2 TO 11
     Y FROM 172 TO 306

PREDICTIONS:
     X FROM 0 TO 13
     Y FROM 0 TO 329.999254
     BASED ON THE PARAMETERS:
          330
          1
          0

GRAPH:
     X FROM 0 TO 13
     Y FROM 0 TO 350
```

Second choice: initial parameters given to MICRO SIMPLEX in Listing 4.1.

Listing 6.1. (Second page) Input to PREDICT merged with the program in Listing 3.1. This leads to a plot of Ricker's cisco length data, along with two predicted curves for which the triplet (y_∞, K, t_0) is (310.8, .3623, -.1898) or (330, 1, 0). A data summary for the second curve is given in the final lines. The graph resulting from this input is shown earlier in Figure 1.1.

7.1 Multiple minima

As a rather artificial example, consider the function

$$F = A \sum_{i=1}^{N} X_i^2 - B \sum_{i=1}^{N} \cos(\pi Y_i), \qquad (7.1)$$

where

$$Y_i = \prod_{j=1}^{i} X_j . \qquad (7.2)$$

In words, Y_i is the product of the first i X's. Notice that the minimum value of $-\cos(Y)$ is -1, achieved when Y is an even integer. It follows that, if $A = 0$ and $B > 0$, then the minimum value of F is $-BN$, achieved when Yi is an even integer ($i = 1, \ldots, N$). It turns out that this implies a proliferation of very close minima as far as the X's are concerned. For example, if $N = 3$, then the following points each correspond to distinct minima:

X_1, X_2, X_3	Y_1, Y_2, Y_3
4, 2, 1	4, 8, 8
4, 2.5, 1	4, 10, 10
4, 2, 1.25	4, 8, 10
4, 2.5, 1.2	4, 10, 12

The larger the X's become, the more dense is the collection of minima.

If $A > 0$ and $B > 0$, then F still has minimum $-BN$, but this is achieved only when all X's are 0. Nevertheless, for small A relative to B, there are still many local minima which could easily trap the search algorithm before it finds the true minimum. The system diskette contains a user program MULTIMIN which codes the function (7.1) - (7.2) according to first level rules (section 3). The inquisitive user is encouraged to experiment with this program merged with MICRO SIMPLEX and PROFILE. Figure 7.1 shows three profiles at F when $A = 1$, $B = 10$, and $N = 3$, as described in Listing 7.1.

Although the example here is quite artificial, for simplicity, fisheries problems include situations of multiple minima with considerable practical interest. For example, the problem of distinguishing age groups of fish by modes in the population length distribution often has no unique solution. It may be possible to account for observed modes in several ways. For a detailed discussion of this problem, see Schnute and Fournier (1980). This paper shows a way to overcome the problem to some extent by re-formulating it a new way. In many cases, that is the practitioner's only recourse. If the problem has too many minima, then it may be necessary to restate it in a more restrictive way, hopefully drawing on information not used before.

```
THIS FUNCTION HAS THE FORM:
F = A * (SUM OF SQUARES OF X'S)
     - B * SUM OF (COS(P1*Y(I)))
WHERE A>=0, B>=0, AND Y(I) IS
THE PRODUCT OF THE FIRST I X'S.

FOR EXAMPLE, WHEN N=2:
F = A*(X1↑2+X2↑2)
     -B*COS(PI*X1)-B*COS(PI*X1*X2)

THE MINIMUM OF F IS -B*N.
IF A=0 AND B>0, THIS IS ACHIEVED
WHEN ALL Y'S ARE EVEN INTEGERS.
IF A>0 AND B>0, THE GLOBAL MIN (-B*N)
OCCURS ONLY WHEN ALL X'S (AND Y'S)
ARE ZERO; HOWEVER OTHER LOCAL MINIMA
EXIST.

IN THIS RUN:
  A = 1,  B = 5

MID-POINT FOR ALL PROFILES:
0   0   0

PROFILE 1 BEGINS AT:
-3.1   0   0
AND ENDS AT:
3.1   0   0

PROFILE 2 BEGINS AT:
-3.1   -3.1   0
AND ENDS AT:
3.1   3.1   0

PROFILE 3 BEGINS AT:
-3.1   -3.1   -3.1
AND ENDS AT:
3.1   3.1   3.1

PROFILE          MINIMUM          MAXIMUM
#1               -14.9968307      4.36528261
#2               -14.9967068      23.0208689
#3               -14.9965834      40.0124305
ALL              -14.9968307      40.0124305
```

Listing 7.1. A description of three profiles for the function (7.1) - (7.2), coded in the file MULTIMIN. Graphs of these profiles are shown in Figure 7.1. Documentation of this function, as shown here, is produced automatically by the initialization phase of the program.

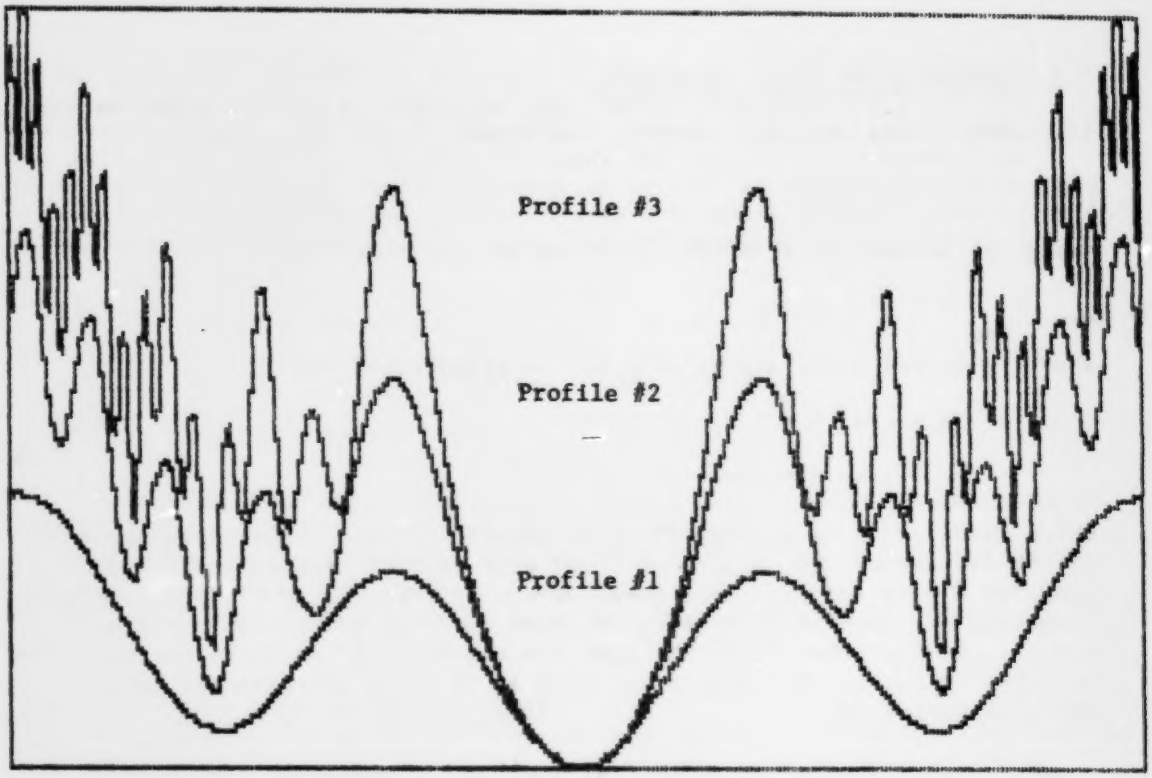


Figure 7.1. Three profiles of the function (7.1) - (7.2). This function is coded in the file MULTIMIN, and the profiles here are described in Listing 7.1. Profiles #1 and #3 are the lower smooth curve and the upper jagged curve, respectively. Profile #2 lies in between.

7.2. Choosing the right parameters

It isn't difficult to see that the right parameters might be very important. Consider, for example, the simple linear relationship

$$y = a + bx \tag{7.3}$$

Using new parameters c and d, (7.3) can be rewritten

$$y = (c + x)/d, \tag{7.4}$$

where (a,b) and (c,d) are related by the transformations

$$a = c/d, \; b = 1/d, \tag{7.5}$$

$$c = a/b, \; d = 1/b. \tag{7.6}$$

Which formulation is better, (7.3) or (7.4)? In many cases there may be little difference, but sometimes (7.4) will lead to disaster. If, for example, $b = -1$ and the first guess is $b = 1$, then, using (7.3), the search algorithm can move in an orderly way past $b = 0$ in going from $b = 1$ to $b = -1$. On the other hand, the same process in (7.4) requires moving d from 1 to $+\infty$ to $-\infty$ to -1. The search may fail as it tries increasingly large positive values of d.

This apparently artificial problem has practical counterparts in real fishery problems. Schnute (1981) shows how a common four-parameter von Bertalanffy model leads to exactly the kind of problem just discussed. The system diskette contains the user program GROWTH, coded by second level rules, which allows a practitioner to do all the analysis described by Schnute (1981). In addition, another program, GROW PARS allows easy transformation between traditional von Bertalanffy parameters and Schnute's parameters. GROW PARS is a complete program, not to be merged with MICRO SIMPLEX or any other program. The use of both these programs is self-documenting, provided the practitioner is familiar with the new model posed in Schnute (1981).

As one final example, consider the common exponential modification of quadratic growth

$$y = c_0 + c_1 x^b + c_2 x^{2b}. \tag{7.7}$$

When $b = 1$ in (7.7), the relation between y and x is parabolic. Another way to write (7.7) is

$$y = y_1 + a_1 z + a_2 z^2 \quad , \tag{7.8}$$

where

$$z = \begin{cases} ((x/x_1)^b - 1)/b, & b \neq 0 \quad , \\ \log(x/x_1), & b = 0 \quad . \end{cases} \tag{7.9}$$

(In (7.9), x_1 is regarded as a number prescribed by the user, or an auxiliary

parameter in the language of section 3.) The quantity y_1 is a parameter representing the value of y when $x = x_1$. Equations (7.7) and (7.8) are equivalent, except that the former uses parameters (c_0, c_1, c_2, b) and the latter uses (y_1, a_1, a_2, b). It turns out for several reasons that (7.8) - (7.9) is a much better formulation of the problem than (7.7) for the purposes of computer search. The system user program REGRESS, coded by second level rules, is designed to find parameter estimates in (7.8) - (7.9).

8. ADDITIONAL POSSIBILITIES

A function minimizing program can be used to solve problems, which, on the surface, are not simple minimization problems. The key lies in reformulating the problem properly. Even in conventional situations, there are sometimes tricks which make the algorithm more flexible and/or more efficient. Some of these possibilities are discussed in this section.

8.1. Arbitrary information functions

Examples cited so far have focussed primarily on the sum of squares, such as S and S* in (1.4) and (1.8), as the measure of information to be minimized. In case the model is used to describe frequency data, the χ^2 function might be used instead. Schnute and Fournier (1980) use a "separation statistic" which has its roots in information theory. See that paper for a full discussion of information functions for frequency data.

In general, statistical models require that the likelihood function be maximized, or, equivalently, that the negative log likelihood be minimized. A very simple example of such a function is given in (3.2), and Listing 3.2 shows the appropriate user program for this case. This example illustrates an obvious, but important, principal: to maximize a function G, minimize the function $F = -G$. When $G > 0$, it is sometimes preferable to use $F = -\log G$.

If the statistical problem is set in a Bayesian framework, then the likelihood is replaced with a posterior probability distribution, equal to the likelihood times a prior distribution. In this case, the practitioner might look for highest posterior density (H.P.D.) estimates, rather than maximum likelihood estimates. With the software here, this involves simply adding a few lines of code, describing the prior distribution, to the code which describes the likelihood function.

8.2. Some parameters held fixed

In some cases the practitioner may build a model with N parameters, but also wish to explore how well it works when certain parameters are set at prescribed values. For instance, one might wish to explore the von

Bertalanffy model (1.1) with $t_0 = 0$. This reduces the three-parameter model (1.1) to its two-parameter counterpart

$$y = y_\infty (1 - e^{-Kt}). \tag{8.1}$$

The software here makes it particularly easy to hold one or more parameters fixed. Listing 8.1 illustrates how to do it in the case just mentioned. Once again, the user program in Listing 3.1 for the von Bertalanffy model and Ricker's cisco length data is merged with MICRO SIMPLEX. The same algorithm data as before (see Listing 4.3) is entered, including the initial value $t_0 = 0$, except that this time the initial step in t_0 is set at 0. As a result, all points on the initial simplex have $t_0 = 0$. Since each later simplex is built from linear combinations of the vertices of the preceding one, it follows by induction that $t_0 = 0$ throughout the run, as listing 8.1 shows.

In short, to hold a parameter fixed at its initial value, simply set the initial step in that parameter to 0.

Listings 4.3 and 8.1 illustrate several typical distinctions between fixed-parameter and free-parameter runs. The respective results are:

y_∞	K	t_0	Minimum S
310.8	.3623	-.1898	46.37
308.4	.3911	0	62.42

The minimum value of S is lower with t_0 free than with t_0 restricted to 0, due to the simple fact that lower values of S are available within a wider range of possible parameters. It take 182 calls and 82 iterations to find the first minimum, but only 125 calls and 67 iterations to find the second, since the latter run involves finding fewer parameters.

Using the minimum value of S in each case, one can perform an analysis of variance to accept or reject the hypothesis that $t_0 = 0$. In this case, there are 10 data points with 3 parameters when t_0 is free or 2 parameters when $t_0 = 0$. The variance ratio

$$\frac{(62.42 - 46.37) / (3 - 2)}{46.37 / (10 - 3)} = 2.42$$

is approximately F-distributed with 1 (3 - 2) and 7 (10 - 3) degrees of freedom. Since

$$F_{.05}\ (1,\ 7) = 5.59$$

is larger than the observed variance ratio, this suggests that the null hypothesis $t_0 = 0$ cannot be rejected at the 5% confidence level. See Schnute (1981) for further details on these methods, particularly regarding growth models.

```
3 VARIABLE(S)
INITIAL POINT:
330    1    0
INITIAL STEP:
50    .2    0
REQUIRED STD. DEV. = 1E-05
MAX. OF 500 FUNCTION CALLS
CONSOLE DISPLAY CODE = 1
PRINTER DISPLAY CODE = 20
CONV. CHK. EACH 1 ITERATIONS
STEP REDUCED BY 1E-03 ON RESTART

INITIAL POINT:
330    1    0
INITIAL FUNCTION:  42076.0222
INITIAL SIMPLEX HI AND LO:
120932.255    42076.0222

37 CALLS, 20 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
305.870715    .410453517    0
CURRENT HI AND LO:
1213.16189    108.058432

72 CALLS, 40 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
308.606467    .390699448    0
CURRENT HI AND LO:
63.1902742    62.5143305

106 CALLS, 60 ITERS., 0 RESTARTS
CURRENT MINIMUM FOUND AT:
308.43824    .391110541    0
CURRENT HI AND LO:
62.4218756    62.4214403

125 CALLS, 67 ITERS., 0 RESTARTS
FINAL POINT PASSED MINIMUM TEST
MINIMUM FOUND AT:
308.439528    .391090213    0
FN. VALUE AT THIS POINT:  62.4213995
```

Listing 8.1. Run of MICRO SIMPLEX merged with the program in Listing 3.1. Here a von Bertalanffy model (1.1) is applied to Ricker's (1975) cisco length data, with the stipulation that $t_0 = 0$. This is achieved by setting $t_0 = 0$ at the initial point and prescribing an initial step of 0 in this parameter. Compare this run with Listing 4.3.

8.3. Some parameters linear

Sometimes the model is nonlinear in certain parameters, but actually linear in others. As a practical example from the fisheries literature, consider a model in which y depends on two variables x_1, and x_2 as follows

$$y = b_0 + b_1 x_1^{\alpha_1} + b_2 x_2^{\alpha_2} + b_{11} x_1^{2\alpha_1} + b_{12} x_1^{\alpha_1} x_2^{\alpha_2} + b_{22} x_2^{2\alpha_2}. \quad (8.2)$$

Notice that (8.2) is an extension in slightly different notation, of the modified quadratic (7.7) discussed earlier. The model (8.2) was used by Bilton et al (1981) to describe the percentage return y of coho adults in response to the time x_1 and size x_2 at which juvenile coho are released.

Notice that (8.2) involves eight parameters: b_0, b_1, b_2, b_{11}, b_{12}, b_{22}, α_1, and α_2; however, it is linear in the first six. If α_1 and α_2 are given, then ordinary linear regression can be used to find all the b's. To be slightly more precise, let S(b's, α's) represent the usual sum of squared model errors. Then, given α_1 and α_2, one can find the b's which minimize S by linear regression of y on

$$x_1^{\alpha_1},\ x_2^{\alpha_2},\ x_1^{2\alpha_1},\ x_1^{\alpha_1} x_2^{\alpha_2},\ \text{and}\ x_2^{2\alpha_2}.$$

Substituting these b's back into S gives

$$F(\alpha_1, \alpha_2) = \min_{b's} S(b's, \alpha's). \quad (8.3)$$

Since F depends on only two variables, rather than eight as S does, the search should go much faster for F than for S. The price paid for this improvement is that the definition of F is complex; it involves performing a six-parameter linear regression. Essentially, this means inverting a 6 × 6 matrix. The data analyis in Bilton et al (1981) was done by this method: a program for F in (8.3) used in conjunction with MICRO SIMPLEX.

To aid the user in solving such problems, the system diskette contains two matrix inversion subroutines. MINV1 inverts without pivotting and MINV2 with pivotting. (Pivotting is slower, but numerically more stable). These subroutines have statement numbers and notation compatible with MICRO SIMPLEX, PROFILE, and PREDICT. They invert a square matrix A of dimension NA. On input, A should contain the matrix to invert. On completion, A contains the inverted matrix, and DA contains the determinant of the original matrix. Here is a short table of approximate performance times in seconds on the Apple II:

NA	Invert without pivot	Invert with pivot
2	.5 sec.	1.0 sec.
3	1.0	2.0
4	1.5	3.5
5	3.0	6.0
6	5.5	9.5
7	8.5	14.0
8	12.0	20.0
9	17.0	27.5
10	23.0	37.5

8.4. Constrained problems

Model parameters are not always free to take arbitrary values. For instance, negative values may be inadmissible, as in the case of the standard deviation in (3.2). An even more restrictive practical example is found in length-frequency analysis, where certain parameters are percentages, or fractions, confined to values between 0 and 1. (See Schnute and Fournier (1980).) All such restrictions are called constraints. They may be used not only to restrict parameters to biologically reasonable values, but also to guide the search algorithm. For example, in case there are multiple minima, constraints can be used to single out a particular minimum of interest.

The software here allows constraints to be added easily, usually by a slight modification of the function F. For instance, if P is a parameter representing a percentage, then the user program might include the two extra lines:

```
IF P < 0 THEN F = F + C * P * P

IF P > 1 THEN F = F + C * (P - 1) * (P - 1)
```

where C is a large constant. The extra terms are called penalties, the price of selecting unacceptable parameter values.

Sometimes constraints are equalities, rather than inequalities as discussed above. For example, P_1, P_2, and P_3 may be percentages whose total is 1:

$$P_1 + P_2 + P_3 = 1. \qquad (8.4)$$

In such cases, the best course is usually to define F in terms of a restricted number of parameters by defining one parameter in terms of others. For example, (8.4) might be handled by the code:

```
P1 = X(1) : P2 = X(2)

P3 = 1 - P1 - P2
```

where F depends on X(1) and X(2).

It is usually a bad idea to replace a constraint like (8.4) with two inequalities, such as

$$P_1 + P_2 + P_3 \leq 1, \tag{8.5}$$

$$P_1 + P_2 + P_3 \geq 1. \tag{8.6}$$

If these two inequalities were built into F with penalty functions, the search would essentially be limited to a narrow region near the constraint (8.4). This makes for inefficient performance. However, sometimes only one of the inequalities, like (8.5) or (8.6), is equivalent to the equality constraint. For example, if

$$F = P1 * P1 + P2 * P2 + P3 * P3, \tag{8.7}$$

then minimizing (8.7) with the constraint (8.4) is the same as minimizing (8.7) with the constraint (8.6). In this case only one penalty function need be used, and the algorithm can perform efficiently. Bilton et al (1981) contains a practical fisheries problem solved by this device.

8.5. Nonlinear equations

Function minimization can also be used to solve non-linear equations. For example, to solve

$$x = a \log x \tag{8.8}$$

for x, given the value of a, define

$$F(x) = (x - a \log x)^2. \tag{8.9}$$

The minimum value of F is 0, achieved at the value of x which solves (8.8). This idea can be extended to cover systems of several equations in several unknowns. In this case F is defined as a sum of squares, with one term for each equation.

REFERENCES

Bilton, H. T., D. F. Alderdice, and J. T. Schnute. 1981. Influence of time and size of release of juvenile coho salmon on returns at maturity. Can. J. Fish. Aquat. Sci. To appear.

Nash, J. C. 1979. Compact numerical methods for computers: linear algebra and function minimization. Halsted Press, John Wiley and Sons (New York): 227 p.

Nelder, J. A., and R. Mead. 1965. A simplex method for function minimization. Computer J. 7: 308-313.

O'Neill, R. 1971. Algorithm AS 47, function minimization using a simplex procedure. Appl. Stat. 20: 338-345.

Ricker, W. E. 1975. Computation and interpretation of biological statistics of fish populations. Fish. Res. Board Can. Bull. 191: 382 p.

Schnute, Jon. 1981. A versatile growth model with statistically stable parameters. Can. J. Fish. Aquat. Sci. To appear.

Schnute, Jon and David Fournier. 1980. A new approach to length-frequency analysis: growth structure. Can. J. Fish. Aquat. Sci. 37: 1337-1351.

Spendley, W., G. R. Hext, and F. W. Himsworth. 1962. Sequential application of simplex designs in optimisation and evolutionary operation. Technometrics 4: 441-461.

APPENDIX A. TECHNICAL REQUIREMENTS AND SUGGESTIONS

A.1. General Requirements

The software described here will operate on an Apple II microcomputer with 48K memory and at least one disk drive operated under DOS 3.3. A printer, preferably in slot 1, is desireable, but not essential. The user must be familiar with APPLESOFT and DOS 3.3, and must know all the usual procedures for operating the system.

The software here requires that programs be merged frequently. For this purpose, I strongly urge the user to acquire the Applesoft Toolkit software sold by Apple Computer Corp. Within this package is the program APA (Applesoft Programmer's Assistant) which contains an extremely fast merge routine. The RENUMBER program on the DOS 3.3 master diskette will do the same job, but much more slowly. Incidentally, the 3.2.1. version of RENUMBER contains serious problems and shouldn't be used. Programs can also be merged by capturing and EXECing text files, but this method is extremely slow and cumbersome.

Personally, I keep the Toolkit files APA, LOADAPA, RLOAD and RBOOT on my simplex diskette, I begin by RUNning LOADAPA, and then keep APA in memory for easy program development and merging. If I have a user program, say GROWTH, which gets used a lot, then I save merged versions of it in files named (in this case) SIMPLEX GROWTH, PROFILE GROWTH, and PREDICT GROWH. In extreme cases, it may be necessary to cancel APA (with the command FP) to make space for data during a program run.

A.2. Graphics Memory Protection

The programs PROFILE and PREDICT use the Apple's hi-resolution graphics page 1. This area of memory needs protection before these programs are loaded and run. On the other hand, the larger program MICRO SIMPLEX doesn't use graphics, but sometimes needs the extra room in the graphics memory area. The simplex disk contains the program PROTECT to make it easy for the user to protect and release this area of memory. This program is self-prompting, and it is automatically invoked at the user's request by the HELLO program. In a typical application, the user would (1) boot the system with the simplex disk, (2) request the use of PROTECT, (3) protect graphics page 1, and (4) run LOADAPA or RENUMBER. As long as programs don't run out of room, the system can be left in this configuration. Otherwise, it may be necessary to use PROTECT to free the graphics area and/or type FP to release APA or RENUMBER.

A.3. Compressed Programs

To allow as much room as possible for the user program, the system diskette contains crunched versions of the main programs, under the names MICRO SIMPLEX.C, PROFILE.C, and PREDICT.C. Normally, the user should always merge these space-saving versions with his or her program. They have been

prepared with Dakin 5 Coporation's "Programming Aids 3.3 Cruncher" and the "&C" command in APA. These combined compressions give a very dense final program. Users with access to APA may also wish to compress user programs with "&C".

A.4. Hard-Copy Graphics

Hard-copy graphics depends, of course, on the particular printer. In this report, all reproductions of the graphics screen have been done on an Integral Data Systems 440 "Paper Tiger" printer, with the aid of software by Computer Station. This particular software works fast and well, and I recommend it to users with Paper Tiger printers. In general, each printer requires its own graphics software. Both PROFILE and PREDICT offer the option of saving the graphics picture on diskette for later hard-copy reproduction.

A.5. Vector List Subroutine

Imbedded in MICRO SIMPLEX, PROFILE, PREDICT, and TRY, is a subroutine to list a vector, like X, of dimension N. It compresses the output in a way which displays as many coordinates as possible on one line. The subroutine is particularly in evidence during output from MICRO SIMPLEX runs. User programs can call this subroutine at statement 59500. It lists the vector V0(N) using W0 columns. Set W0 = 39 for the screen and W0 = 79 for the printer.

A.6. Data Files

The user program typically involves entry of observed data. My philosophy here is to use a general-purpose editor for preparing and maintaining data files, and to write the user program so that data comes from a file. I have been using a filing program called COMPUSHOP DATABASE available through the Compushop in Vancouver, B. C. This program stores data in sequential files, organized by records and fields within records. If, for example, the file has M logical records with K fields in each, then COMPUSHOP DATABASE creates a sequential text file with K(M + 1) + 3 physical records. The first record contains the number K. The next K records contain the title (a string) for each field. The next record contains the number M, and the next contains the approximate number of bytes per logical record. This makes K + 3 physical records so far. After that, there are M sets of K physical records representing the M logical records.

The incomplete user program TEMPLATE (section 3) assumes that data are stored on a file in the above format with K = 2. Within that file, there are N0 logical records each containing the two fields X0 and Y0. The user who wishes to include TEMPLATE in his or her program must generate files in this format, or modify TEMPLATE accordingly. Incidentally, the program EDIT in the Applesoft Toolkit can be used, with a bit of careful bookkeeping, to generate the necessary data files. If available, COMPUSHOP DATABASE works quite well on files of the size typical in MICRO SIMPLEX applications. The system diskette includes the samples files LENGTH DATA and WEIGHT DATA in the required format. These contain Ricker's (1975, p.226) length and weight data for ciscoes, used in many examples in this report.

APPENDIX B. VERSION 1.1 FILES

File	Section(s) where discussed
GROWTH	7.2
GROWTH PARS	7.2
HELLO	App. A.2
LENGTH DATA	App. A.6
MICRO SIMPLEX	4
MICRO SIMPLEX.C	App. A.3
MINV 1	8.3
MINV 2	8.3
MULTIMIN	7.1
PREDICT	6
PREDICT.C	App. A.3
PROFILE	5
PROFILE.C	App. A.3
PROTECT	5, App. A.2
REGRESS	7.2
TEMPLATE	3, App. A.6
TRY	3
WEIGHT DATA	App. A.6

APPENDIX C. WORKED EXAMPLE

1. Boot the system using the simplex diskette. Press "space" a few times and note the descriptive list of programs on the diskette. Respond "Y" to the option to run PROTECT, "Y" to graphics protection, and protect page "1".

2. If you have the Applesoft Toolkit, RUN LOADAPA. Otherwise, RUN RENUMBER on the DOS 3.3 Master diskette. When you're finished, make sure the drive with the simplex diskette is re-selected.

3. Type NEW. Then enter the two lines:

```
3100 YI = X(1) : K = X(2) : TZ = X(3) : T = XP

3110 YP = YI * (1 - EXP (-K * (T - TZ)))
```

4. Type "&H". Then "LOAD TEMPLATE", then "&M", then "SAVE VON B". At this point you have a complete user program, according to second level rules (section 3), saved in a file called VON B. It describes the von Bertalanffy growth model (1.1).

5. Type "&H". Then "LOAD MICRO SIMPLEX.C", then "&M". You have now combined VON B with MICRO SIMPLEX.

6. Type "RUN". Turn to the two-page Listing 4.1 and follow all the steps shown there. If you have no printer or if your printer isn't connected to slot #1, then respond "0" to the prompt "ITERATIONS BET. PRINTER DISPLAYS", rather than "20" as shown.

7. Having reached the end of Listing 4.1 (second page), respond "Y" to "EVERYTHING OK?".

8. In response to a question, type "A" to indicate additive errors. Also reply "LENGTH DATA" to the prompt for an observed data file. Indicate "S" to list the data on screen.

9. After the data are listed, you should see Listing 4.2 emerging on the screen and Listing 4.3 appearing slowly on the printer. Watch patiently. MICRO SIMPLEX will gradually locate optimal parameters y_∞, K, and t_o for LENGTH DATA. When the process is complete, reply "N" to the possibility of starting again from the final point.

10. Begin again by "LOAD VON B", "&H", "LOAD PREDICT.C", and "&M". You have now combined VON B with PREDICT.

11. Type "RUN". Turn to the two-page Listing 6.1 and follow the steps shown there. After responding "B" for BOTH, you will be asked the name of the observed data file. As in step 8, reply "LENGTH DATA". Indicate "S" for a list of data on the screen.

12. Continue with Listing 6.1. There will be a pause after specifying "70" points on the curve, while the predicted curve is calculated. At a certain point you will see part of Figure 1.1 on the screen. Reply "Y" to the question about a printed data summary if you have a printer connected to slot #1; otherwise "N".

13. After entering the second set of parameters, there will be another pause for calculation, and the completed Figure 1.1 should appear on the screen. Again, request a printer summary, if appropriate.

14. To exit PREDICT, press the "RESET" key.

15. Begin once more by "LOAD VON B", "&H", "LOAD PROFILE.C", and "&M". You have now combined VON B with PROFILE.

16. Type "RUN". Turn to the two-page Listing 5.1 and follow the responses shown there. As in steps 8 and 11, "LENGTH DATA" is the observed data file. Again, the response "S" gives a list on the screen.

17. There will be a very long pause at this point while the requested profiles are calculated. The Apple will sound a "beep" when it is finished. At this point you can (and should) request a printed summary if you have a printer in slot #1. The summary should agree with the second page of Listing 5.1.

18. Next, a blank screen with a frame around it should be visible. respond "P" for PLOT, and then plot profile number "1". Similarly, plot profiles "2" and "3". You should see Figure 1.2 emerge in stages on the screen. The three profiles correspond to changes in y_∞, K, and t_o, respectively. See the text accompanying Figure 1.2 for further details.

19. At this point, try the "E" option for ERASE, and then plot, say, variable 2 or variable 3. It should become clear that any one, two, or three of the profiles can be displayed at once. When you're finished, type "Q" for QUIT.

APPENDIX D. PROGRAM LISTINGS

This Appendix presents listings of all Applesoft programs (before compression) on the simplex version 1.1 diskette. In addition, for each program there is an alphabetical list of the variables used, along with the line numbers in which they are referenced. All these listings have been produced with the LISTER and VARIABLE CROSS REFERENCE utilities in Dakin 5 Corporation's Programming Aids 3.3.

```
]
NAME..GROWTH   DATE..03/13/81   TIME..1700   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1000  REM  GROWTH FEB 6/81
1001  REM  REV MAR 02/81
1004  REM  MODULE FOR MICRO SIMPLEX
1005  REM  FOR FITTING SCHNUTE'S GROWTH MODEL TO OBSERVED A
     GE-LENGTH DATA
1010  IF NF > 0 THEN 5000: REM  CALCULATE SUM OF SQUARES OF
      ERRORS
1020  IF NF = 0 THEN 6000: REM  INITIALIZE
1030  IF NF < 0 THEN 7000: REM  FINAL OUTPUT
1100  RETURN
1500  REM  *** PREDICT ENTRY POINT ***
1510  IF NP = 1 THEN 3000: REM  PREDICT YP FROM XP
1520  IF NP =  - 1 THEN 2500: REM  SET UP MODEL PARAMETERS
1530  IF NP =  - 2 THEN 2000: REM  GET OBSERVED DATA
1540  RETURN
2000  REM  *** GET OBSERVED DATA FROM FILE ***
2030  PRINT "NAME OF THE OBSERVED DATA FILE?"
2040  INPUT FI$
2050 D$ =  CHR$ (4):AZERO = .0001:BZERO = .02
2070  PRINT : PRINT "READING DATA . . ."
2100  PRINT D$;"OPEN";FI$
2110  PRINT D$;"READ";FI$
2120  INPUT NI
2130  IF NI <  > 2 THEN  PRINT NI;" ITEMS/RECORD. RIGHT FIL
     E?": PRINT D$;"CLOSE": STOP
2135  INPUT W$,W$: REM  SKIP ITEM NAMES
2140  INPUT NO
2150  INPUT W: REM  SKIP BYTES/RECORD
2200  IF RN = 0 THEN  DIM XO(NO),YO(NO)
2210 RN = 1
2220  FOR I = 1 TO NO
2230  INPUT XO(I),YO(I)
2240  NEXT
2250  PRINT D$;"CLOSE"
2300  PRINT "SHOW DATA ON SCREEN?"
2310  INPUT "Y OR N? ";R$
2320  IF R$ <  > "Y" THEN 2400
2330  PRINT "AGE","SIZE"
2340  FOR I = 1 TO NO
2350  PRINT XO(I),YO(I)
2360  NEXT
2400  RETURN
2500  REM  *** SET UP MODEL PARAMETERS ***
2510  PRINT : PRINT "GROWTH MODEL WITH PARAMETERS"
2520  PRINT "Y1, Y2, A, AND B.": PRINT
2530  INPUT "WHAT IS T1? ";T1
2540  INPUT "WHAT IS T2? ";T2
2600  RETURN
3000  REM  *** PREDICT YP FROM XP ***
3010 Y1 = X(1):Y2 = X(2):A = X(3):B = X(4)
3020 AA =  ABS (A):AB =  ABS (B)
```

```
NAME..GROWTH   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

3030  IF (AA > AZ AND AB > BZ) THEN 3200
3040  IF (AA > AZ AND AB <  = BZ) THEN 3400
3050  IF (AA <  = AZ AND AB > BZ) THEN 3600
3060  IF (AA <  = AZ AND AB <  = BZ) THEN 3800
3200 Z1 = Y1 ↑ B:Z2 = Y2 ↑ B
3210 W1 = 1 -  EXP ( - A * (XP - T1))
3220 W2 = 1 -  EXP ( - A * (T2 - T1))
3230 YP = (Z1 + (Z2 - Z1) * W1 / W2) ↑ (1 / B)
3240  RETURN
3400 W1 = 1 -  EXP ( - A * (XP - T1))
3410 W2 = 1 -  EXP ( - A * (T2 - T1))
3420 YP = Y1 *  EXP ( LOG (Y2 / Y1) * W1 / W2)
3430  RETURN
3600 Z1 = Y1 ↑ B:Z2 = Y2 ↑ B
3610 YP = (Z1 + (Z2 - Z1) * (XP - T1) / (T2 - T1)) ↑ (1 / B)
3620  RETURN
3800 W =  LOG (Y2 / Y1)
3810 YP = Y1 *  EXP (W * (XP - T1) / (T2 - T1))
3820  RETURN
5000  REM  *** SUM OF SQUARES OF ERRORS ***
5010 S = 0
5020  FOR I = 1 TO NO
5030 XP = XO(I)
5040  GOSUB 3000: REM  PREDICT YP FROM XP
5100  IF E$ = "A" THEN ERR = YO(I) - YP
5110  IF E$ = "L" THEN ERR =  LOG (YO(I) / YP)
5120 S = S + ERR * ERR
5130  NEXT I
5140 F = S
5200  RETURN
6000  REM  *** INITIALIZE ***
6010  GOSUB 2500: REM  SET UP MODEL PARAMETERS
6100  PRINT "ERRORS ADDITIVE OR LOGARITHMIC?"
6110  INPUT "A OR L? "$E$: REM  ERROR STRUCTURE
6120  IF  NOT (E$ = "A" OR E$ = "L") THEN 6100
6130  PRINT
6200  GOSUB 2000: REM  GET OBSERVED DATA
6210  RETURN
7000  REM  *** FINAL OUTPUT ***
7010  PRINT "DO YOU WANT AN EXTENDED LIST"
7020  PRINT "   OF PARAMETERS?"
7030  INPUT "Y OR N? "$R$
7040  IF R$ = "N" THEN  RETURN
7100  GOSUB 8000
7110  GOSUB 9000
7120  GOSUB 7500
7130  PRINT "PARAMETERS ON PRINTER?"
7140  INPUT "Y OR N? "$R$
7150  IF R$ = "N" THEN  RETURN
7200 SL = 1
7210  PRINT D$$"PR#"$SL
```

```
NAME..GROWTH   DATE..03/13/81   TIME..1600   PAGE..03
STARTING LINE..0    ENDING LINE..63999

7220  GOSUB 9000
7230  GOSUB 7500
7240  PRINT D$;"PR#0"
7250  RETURN
7500  IF E$ = "A" THEN CAP$ = "ADD. "
7510  IF E$ = "L" THEN CAP$ = "LOG. "
7520 DF = NO - 4
7530 SD =  SQR (T  / DF )
7540  PRINT CAP$;"VARIANCE",T
7550  PRINT CAP$;"STD. ERR.",SD
7560  PRINT "DEG. FREEDOM",DF
7570  PRINT "         WITH",NO;"  DATA POINTS"
7600  RETURN
8000  REM  COMPUTE VB PARS
8010 U1 =  EXP (A * T1)
8020 U2 =  EXP (A * T2)
8030 W1 = Y1 ↑ B
8040 W2 = Y2 ↑ B
8050 U3 = U2 - U1
8060 W3 = W2 - W1
8070 UW = U2 * W2 - U1 * W1
8080 UL = U2 *  LOG (Y2) - U1 *  LOG (Y1)
8100  IF A = 0 THEN C = T2 - T1: GOTO 8120
8110 C = (1 / U1 - 1 / U2) / A
8120  IF B = 0 THEN D =  LOG (Y2 / Y1): GOTO 8150
8130 D = (W2 - W1) / B
8150 Z1 = D / (C * U1 * W1)
8160 Z2 = D / (C * U2 * W2)
8200 TO = 0: IF B = 0 THEN 8300
8205  IF A <  > 0 THEN 8230
8210 TO = T1 + T2 - (T2 * W2 - T1 * W1) / W3
8220  GOTO 8300
8230 W = UW / W3
8240  IF W <  = 0 THEN TO = 0: GOTO 8300
8250 TO = T1 + T2 -  LOG (W) / A
8300 Y1 = 0: IF A = 0 THEN 8400
8305  IF B <  > 0 THEN 8330
8310 Y1 =  EXP (UL / U3)
8320  GOTO 8400
8330 W = UW / U3
8340  IF W <  = 0 THEN Y1 = 0: GOTO 8400
8350 Y1 = W ↑ (1 / B)
8400 TS = 0: IF A = 0 THEN 8500
8405  IF B <  > 0 THEN 8430
8410 TS = T1 + T2 -  LOG (U3 /  LOG (Y2 / Y1)) / A
8420  GOTO 8500
8430 W = B * UW / W3
8440  IF W <  = 0 THEN TS = 0: GOTO 8500
8450 TS = T1 + T2 -  LOG (W) / A
8500 YS = 0: IF A = 0 THEN 8600
8505  IF B <  > 0 THEN 8530
```

```
NAME..GROWTH   DATE..03/13/81   TIME..1600   PAGE..04
STARTING LINE..0    ENDING LINE..63999

8510 YS =  EXP (UL / U3 - 1)
8520  GOTO 8600
8530 W = (1 - B) * UW / U3
8540  IF W <  = 0 THEN YS = 0: GOTO 8600
8550 YS = W ↑ (1 / B)
8600 ZS = 0
8610  IF B <  > 1 THEN ZS = A / (1 - B)
8620 AB =  - B *  LOG (Y2 / Y1) / (T2 - T1)
8700 RE = 0
8710  IF (A > 0 AND B < 1 AND B > 0) THEN RL = 1
8720  IF (A > 0 AND B >  = 1) THEN RL = 2
8730  IF (AB < A AND A <  = 0 AND 1 < B) THEN RL = 3
8740  IF (AB < A AND A <  = 0 AND B >  = 0 AND B <  = 1) THEN RL =
    4
8750  IF (A <  = AB AND B >  = 0) THEN RL = 5
8760  IF (A < 0 AND B < 0) THEN RL = 6
8770  IF (A >  = 0 AND A <  = AB AND B < 0) THEN RL = 7
8780  IF (AB < A AND B <  = 0) THEN RL = 8
8800  RETURN
9000  REM  LIST PARS
9010  PRINT
9020  PRINT "T1",T1
9030  PRINT "T2",T2
9100  PRINT "A",A
9110  PRINT "B",B
9120  PRINT "Y1",Y1
9130  PRINT "Y2",Y2
9140  PRINT "Z1",Z1
9150  PRINT "Z2",Z2
9200  PRINT "Y INITINITY",YI
9210  PRINT "T ZERO",T0
9220  IF B = 0 THEN 9230
9222 P = 1 / B:Q =  - P
9224  PRINT "VB P",P
9226  PRINT "RICHARDS P",Q
9230  PRINT "G",A
9300  PRINT "Y STAR",YS
9310  PRINT "T STAR",TS
9320  PRINT "Z STAR",ZS
9330  PRINT "REGION",RL
9340  PRINT
9400  RETURN
```

** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...GROWTH
DATE...03/13/81

A
3010 3020 3210 3220 3400 3410 8010 8020 8100 8110 8205 8250 8300 8400
8410 8450 8500 8610 8710 8720 8730 8730 8740 8740 8750 8760 8770 8770
8780 9100 9230

AA
3020 3030 3040 3050 3060

AB
3020 3030 3040 3050 3060 8620 8730 8740 8750 8770 8780

AZ
3030 3040 3050 3060

AZERO
2050

B
3010 3020 3200 3200 3230 3600 3600 3610 8030 8040 8120 8130 8200 8305
8350 8405 8430 8505 8530 8550 8610 8610 8620 8710 8710 8720 8730 8740
8740 8750 8760 8770 8780 9110 9220 9222

BZ
3030 3040 3050 3060

BZERO
2050

C
8100 8110 8150 8160

CAP$
7500 7510 7540 7550

D
8120 8130 8150 8160

D$
2050 2100 2110 2130 2250 7210 7240

DF
7520 7530 7560

E$
5100 5110 6110 6120 6120 7500 7510

ERR
5100 5110 5120 5120

F
5140 7530 7540

FI$
2040 2100 2110

I
2220 2230 2230 2340 2350 2350 5020 5030 5100 5110 5130

NF
1010 1020 1030

NI
2120 2130 2130

NO
2140 2200 2200 2220 2340 5020 7520 7570

NP
1510 1520 1530

P
9222 9222 9224

Q
9222 9226

```
R$
2310 2320 7030 7040 7140 7150

RE
8700 8710 8720 8730 8740 8750 8760 8770 8780 9330

RN
2200 2210

S
5010 5120 5120 5140

SD
7530 7550

SL
7200 7210

T0
8200 8210 8240 8250 9210

T1
2530 3210 3220 3400 3410 3610 3610 3810 3810 8010 8100 8210 8210 8250
8410 8450 8620 9020

T2
2540 3220 3410 3610 3810 8020 8100 8210 8210 8250 8410 8450 8620 9030

TS
8400 8410 8440 8450 9310

U1
8010 8050 8070 8080 8110 8150

U2
8020 8050 8070 8080 8110 8160

U3
8050 8310 8330 8410 8510 8530

UL
8080 8310 8510

UW
8070 8230 8330 8430 8530

W
2150 3800 3810 8230 8240 8250 8330 8340 8350 8430 8440 8450 8530 8540
8550

W$
2135 2135

W1
3210 3230 3400 3420 8030 8060 8070 8130 8150 8210

W2
3220 3230 3410 3420 8040 8060 8070 8130 8160 8210

W3
8060 8210 8230 8430

X(
3010 3010 3010 3010

XO(
2200 2230 2350 5030

XP
3210 3400 3610 3810 5030

Y1
3010 3200 3420 3420 3600 3800 3810 8030 8080 8120 8410 8620 9120

Y2
3010 3200 3420 3600 3800 8040 8080 8120 8410 8620 9130

YI
```

```
8300 8310 8340 8350 9200

YO(
2200 2230 2350 5100 5110

Y1'
3230 3420 3610 3810 5100 5110

YS
8500 8510 8540 8550 9300

Z1
3200 3230 3230 3600 3610 3610 8150 9140

Z2
3200 3230 3600 3610 8160 9150

ZS
8600 8610 9320
```

```
]
NAME..GROW PARS   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  GROW PARS JAN 22/81
2  REM  REV JAN 23/81
50 D$ =  CHR$ (4)
100  GOSUB 1000: REM  PRIMARY INPUT
110  GOSUB 2000: REM  SECONDARY INPUT
120  IF MO = 1 THEN  GOSUB 7000: REM  VB TO SCH PARS
125  IF MO = 2 THEN  GOSUB 6000: REM  RICHARDS TO SCH PARS
130  GOSUB 8000: REM   FIND VB PARS
200  GOSUB 9000: REM  PARS ON SCREEN
210  PRINT D$;"PR#1"
220  PRINT
230  GOSUB 9000: REM  PARS ON PRINTER
240  PRINT D$;"PR#0"
300  GOTO 110
1000  REM  PRIMARY INPUT
1010  HOME : VTAB 6
1020  PRINT "INPUT FROM WHICH MODEL?"
1030  PRINT "      1.GENERALIZED VON BERTALANFFY"
1035  PRINT "      2.RICHARDS"
1040  PRINT "      3.SCHNUTE"
1045  PRINT
1050  INPUT MO
1060  IF (MO <  > 1 AND MO <  > 2 AND MO <  > 3) THEN 1000
1070  PRINT
1100  INPUT "T1?    ";T1
1110  INPUT "T2?    ";T2
1200  RETURN
2000  REM  SECONDARY INPUT
2010  IF  MO = 3 THEN 2200
2100  INPUT "YINF ? ";Y1
2110  INPUT "G?     ";G
2120  IF  MO = 1 THEN  INPUT "T0?    ";T0
2125  IF  MO = 2 THEN  INPUT "TSTR? ";TS
2130  INPUT "P?     ";P
2140  IF  MO = 2 THEN Q = P:P =  - P
2150  RETURN
2200  INPUT "Y1?    ";Y1
2210  INPUT "Y2?    ";Y2
2220  INPUT "A?     ";A
2230  INPUT "B?     ";B
2240  RETURN
6000  REM  RICHARDS TO SCH PARS
6010 T = T1: GOSUB 6500:Y1 = Y
6020 T = T2: GOSUB 6500:Y2 = Y
6030 A = G
6040 B = 1 / P
6050  RETURN
6500 W =  EXP ( - G * (T - TS))
6510 Y = Y1 * (1 + W / Q) ^ ( - Q)
6520  RETURN
7000  REM  VB TO SCH PARS
```

```
NAME..GROW PARS    DATE..03/13/81    TIME..1600    PAGE..02
STARTING LINE..0     ENDING LINE..63999

7010 T = T1: GOSUB 7500:Y1 = Y
7020 T = T2: GOSUB 7500:Y2 = Y
7030 A = G
7040 B = 1 / P
7050  RETURN
7500 W =  EXP ( - G * (T - T0))
7510 Y = Y1 * (1 - W) ↑ P
7520  RETURN
8000  REM  COMPUTE VB PARS
8010 U1 =  EXP (A * T1)
8020 U2 =  EXP (A * T2)
8030 W1 = Y1 ↑ B
8040 W2 = Y2 ↑ B
8050 U3 = U2 - U1
8060 W3 = W2 - W1
8070 UW = U2 * W2 - U1 * W1
8080 UL = U2 *  LOG (Y2) - U1 *  LOG (Y1)
8100  IF A = 0 THEN C = T2 - T1: GOTO 8120
8110 C = (1 / U1 - 1 / U2) / A
8120  IF B = 0 THEN D =  LOG (Y2 / Y1): GOTO 8150
8130 D = (W2 - W1) / B
8150 Z1 = D / (C * U1 * W1)
8160 Z2 = D / (C * U2 * W2)
8200 T0 = 0: IF B = 0 THEN 8300
8205  IF A <  > 0 THEN 8230
8210 T0 = T1 + T2 - (T2 * W2 - T1 * W1) / W3
8220  GOTO 8300
8230 W = UW / W3
8240  IF W <  = 0 THEN T0 = 0: GOTO 8300
8250 T0 = T1 + T2 -  LOG (W) / A
8300 Y1 = 0: IF A = 0 THEN 8400
8305  IF B <  > 0 THEN 8330
8310 Y1 =  EXP (UL / U3)
8320  GOTO 8400
8330 W = UW / U3
8340  IF W <  = 0 THEN Y1 = 0: GOTO 8400
8350 Y1 = W ↑ (1 / B)
8400 TS = 0: IF A = 0 THEN 8500
8405  IF B <  > 0 THEN 8430
8410 TS = T1 + T2 -  LOG (U3 /  LOG (Y2 / Y1)) / A
8420  GOTO 8500
8430 W = B * UW / W3
8440  IF W <  = 0 THEN TS = 0: GOTO 8500
8450 TS = T1 + T2 -  LOG (W) / A
8500 YS = 0: IF A = 0 THEN 8600
8505  IF B <  > 0 THEN 8530
8510 YS =  EXP (UL / U3 - 1)
8520  GOTO 8600
8530 W = (1 - B) * UW / U3
8540  IF W <  = 0 THEN YS = 0: GOTO 8600
8550 YS = W ↑ (1 / B)
```

```
NAME..GROW PARS    DATE..03/13/81    TIME..1600    PAGE..03
STARTING LINE..0     ENDING LINE..63999

8600 ZS = 0
8610  IF B <  > 1 THEN ZS = A / (1 - B)
8620 AB =  - B *  LOG (Y2 / Y1) / (T2 - T1)
8700 RL = 0
8710  IF (A > 0 AND B < 1 AND B > 0) THEN RL = 1
8720  IF (A > 0 AND B >  = 1) THEN RL = 2
8730  IF (AB < A AND A <  = 0 AND 1 < B) THEN RL = 3
8740  IF (AB < A AND A <  = 0 AND B >  = 0 AND B <  = 1) THEN RL =
    4
8750  IF (A <  = AB AND B >  = 0) THEN RL = 5
8760  IF (A < 0 AND B < 0) THEN RL = 6
8770  IF (A >  = 0 AND A <  = AB AND B < 0) THEN RL = 7
8780  IF (AB < A AND B <  = 0) THEN RL = 8
8800  RETURN
9000  REM  LIST PARS
9010  PRINT
9020  PRINT "T1",T1
9030  PRINT "T2",T2
9100  PRINT "A",A
9110  PRINT "B",B
9120  PRINT "Y1",Y1
9130  PRINT "Y2",Y2
9140  PRINT "Z1",Z1
9150  PRINT "Z2",Z2
9200  PRINT "Y INFINITY",YI
9210  PRINT "T ZERO",T0
9220  IF  B = 0 THEN 9230
9222 P = 1 / B:Q =  - P
9224  PRINT "VB P",P
9226  PRINT "RICHARDS P",Q
9230  PRINT "G",A
9300  PRINT "Y STAR",YS
9310  PRINT "T STAR",TS
9320  PRINT "Z STAR",ZS
9330  PRINT "REGION",RL
9340  PRINT
9400  RETURN
```

** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...GROW PARS
DATE...03/13/81

A
2220 6030 7030 8010 8020 8100 8110 8205 8250 8300 8400 8410 8450 8500
8610 8710 8720 8730 8730 8740 8740 8750 8760 8770 8770 8780 9100 9230

AB
8620 8730 8740 8750 8770 8780

B
2230 6040 7040 8030 8040 8120 8130 8200 8305 8350 8405 8430 8505 8530
8550 8610 8610 8620 8710 8710 8720 8730 8740 8740 8750 8760 8770 8780
9110 9220 9222

C
8100 8110 8150 8160

D
8120 8130 8150 8160

D$
50 210 240

G
2110 6030 6500 7030 7500

MO
120 125 1050 1060 1060 1060 2010 2120 2125 2140

P
2130 2140 2140 2140 6040 7040 7510 9222 9222 9224

Q
2140 6510 6510 9222 9226

RE
8700 8710 8720 8730 8740 8750 8760 8770 8780 9330

T
6010 6020 6500 7010 7020 7500

TO
2120 7500 8200 8210 8240 8250 9210

T1
1100 6010 7010 8010 8100 8210 8210 8250 8410 8450 8620 9020

T2
1110 6020 7020 8020 8100 8210 8210 8250 8410 8450 8620 9030

TS
2125 6500 8400 8410 8440 8450 9310

U1
8010 8050 8070 8080 8110 8150

U2
8020 8050 8070 8080 8110 8160

U3
8050 8310 8330 8410 8510 8530

UL
8080 8310 8510

UW
8070 8230 8330 8430 8530

W
6500 6510 7500 7510 8230 8240 8250 8330 8340 8350 8430 8440 8450 8530
8540 8550

W1
8030 8060 8070 8130 8150 8210

W2

8040 8060 8070 8130 8160 8210

W3
8060 8210 8230 8430

Y
6010 6020 6510 7010 7020 7510

Y1
2200 6010 7010 8030 8080 8120 8410 8620 9120

Y2
2210 6020 7020 8040 8080 8120 8410 8620 9130

YI
2100 6510 7510 8300 8310 8340 8350 9200

YS
8500 8510 8540 8550 9300

Z1
8150 9140

Z2
8160 9150

ZS
8600 8610 9320

```
]
NAME..HELLO   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  HELLO AUG 13/80
2  REM  REV MAR 12/81
100  GOSUB 1000: REM  MAIN HEADER
120  GOSUB 2000: REM  GET FILE DESCRIPTIONS
140  GOSUB 2500: REM  LIST FILES
160  GOSUB 5000: REM  RUN PROTECT
180  END
900  REM  ON TO NEXT PAGE
910  PRINT : PRINT "PRESS ANY KEY TO CONTINUE . . . ";
920  GET R$
930  PRINT : HOME
940  RETURN
1000  REM  PRINT MAIN HEADER
1010  HOME
1020  PRINT "A SUITE OF PROGRAMS RELATED TO"
1030  PRINT "THE SIMPLEX METHOD OF FUNCTION"
1040  PRINT "MINIMIZATION."
1050  PRINT : PRINT "VERSION 1.1": PRINT
1060  PRINT "SEPARATE DOCUMENTATION IS BEING"
1070  PRINT "PREPARED. IT WILL PROBABLY TAKE"
1080  PRINT "A WHILE."
1090  PRINT
1100  PRINT "  BY JON SCHNUTE"
1110  PRINT "      PACIFIC BIOLOGICAL STATION"
1120  PRINT "      NANAIMO, B.C. V9R 5K6"
1130  PRINT "      CANADA"
1140  PRINT
1200  GOSUB 900: REM  PAGE
1210  RETURN
2000  REM  GET FILE DESCRIPTIONS
2010 N = 12: REM  NO. OF FILES
2020  DIM NA$(N),DE$(2 * N)
2030  FOR I = 1 TO N: READ NA$(I): NEXT
2040  FOR I = 1 TO N
2050  READ DE$(2 * I - 1),DE$(2 * I)
2060  NEXT
2100  RETURN
2500  REM  LIST FILE DESCRIPTIONS
2510 NP = 6: REM  NAMES/PAGE
2520 J = 0
2530  FOR I = 1 TO N
2540  PRINT I;".";NA$(I)
2550  PRINT DE$(2 * I - 1): PRINT DE$(2 * I)
2560 J = J + 1
2600  IF  J = NP THEN J = 0: GOSUB 900: REM  PAGE
2610  NEXT
2620  IF  J > 0 THEN  GOSUB 900: REM  PAGE
2630  RETURN
5000  REM  RUN PROTECT, IF DESIRED
5010  PRINT "MINIMIZATION PROGRAMS WITH 'MICRO"
5020  PRINT "SIMPLEX' ARE SOMETIMES LARGE, AND NEED"
```

```
NAME..HELLO   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

5030  PRINT "TO OVER-WRITE THE GRAPHICS PAGES."
5040  PRINT "OTHER PROGRAMS,HOWEVER, USE GRAPHICS."
5050  PRINT "A PROGRAM 'PROTECT' ALLOWS SIMPLE"
5060  PRINT "PROTECTION AND RELEASE OF  GRAPHICS"
5100  PRINT "PAGES. DO YOU WISH TO RUN 'PROTECT'"
5110  PRINT "NOW?"
5120  PRINT : INPUT "Y OR N? "$R$
5130 D$ =  CHR$ (4)
5140  IF  R$ = "Y" THEN  PRINT D$;"RUN PROTECT"
5200  RETURN
8000  DATA  "MICRO SIMPLEX"
8020  DATA  "PROFILE"
8040  DATA  "PREDICT"
8060  DATA  "TRY"
8080  DATA  "PROTECT"
8100  DATA  "MULTIMIN"
8120  DATA  "TEMPLATE"
8140  DATA  "REGRESS"
8160  DATA  "GROWTH"
8180  DATA  "GROW PARS"
8200  DATA  "LENGTH DATA, WEIGHT DATA"
8220  DATA  "MINV1, MINV2"
9000  DATA  "   PROGRAM TO MINIMIZE A FUNCTION,"
9020  DATA   "   SUCH AS NEGATIVE LOG LIKELIHOOD"
9040  DATA  "   PROGRAM TO PLOT THE FUNCTION, AS"
9060  DATA  "   THE PARAMETERS (COORDINATES) VARY"
9080  DATA  "   PROGRAM TO PLOT THE OBSERVATIONS"
9100  DATA  "   AND A PREDICTED CURVE"
9120  DATA  "   PROGRAM TO DEBUG A USER MODULE"
9140  DATA   "   BY TRYING IT"
9160  DATA  "   PROGRAM TO PROTECT (OR FREE) MEMORY"
9180  DATA  "   SPACE FOR THE GRAPHICS PAGES"
9200  DATA  "   EXAMPLE MODULE FOR A FUNCTION WITH"
9220  DATA  "   MULTIPLE MINIMA"
9240  DATA  "   TEMPLATE FOR CONSTRUCTING A USER"
9260  DATA  "   FUNCTION MODULE"
9280  DATA  "   EXAMPLE MODULE FOR MODIFIED"
9300  DATA   "   QUADRATIC REGRESSION"
9320  DATA  "   EXAMPLE MODULE TO ESTIMATE PAR'S"
9340  DATA  "   FOR SCHNUTE'S GROWTH MODEL"
9360  DATA  "   PROGRAM TO TRANSFORM BETWEEN VON"
9380  DATA   "   BERTALANFFY AND SCHNUTE PARAMETERS"
9400  DATA  "   SAMPLE DATA FOR 'GROWTH' AND/OR"
9420  DATA  "   'REGRESS', RICKER'S HNDBK. P. 226"
9440  DATA  "   SUBROUTINES TO INVERT A MATRIX"
9460  DATA  "   (1) WITHOUT AND (2) WITH PIVOT"
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..03
STARTING LINE..0    ENDING LINE..63999

41410  INPUT I5
41420  PRINT "STEP REDUCTION FRACTION FOR RESTART?"
41430  INPUT S5
41450  RETURN
42000  REM  ENTER DATA ITEMS ****
42010  PRINT "ENTER CODE FOR CHANGED DATA."
42020  PRINT "CD - GIVES LIST OF CODES."
42030  PRINT "NO - NO MORE CHANGES."
42040  INPUT A0$
42100  IF A0$ = "IP" THEN  GOSUB 42500
42110  IF A0$ = "IS" THEN  GOSUB 42600
42120  IF A0$ = "SD" THEN  INPUT R5
42130  IF A0$ = "MI" THEN  INPUT M0
42140  IF A0$ = "CO" THEN  INPUT C0
42150  IF A0$ = "PR" THEN  INPUT P0
42160  IF A0$ = "CC" THEN  INPUT I5
42170  IF A0$ = "SR" THEN  INPUT S5
42200  IF A0$ = "CD" THEN  GOSUB 42700
42210  IF A0$ = "NO" THEN  RETURN
42300  GOTO 42000
42500  PRINT "NEW INITIAL POINT:"
42510  FOR I9 = 1 TO N: INPUT X0(I9): NEXT
42520  RETURN
42600  PRINT "NEW INITIAL STEP:"
42610  FOR I9 = 1 TO N: INPUT S0(I9): NEXT
42620  RETURN
42700  PRINT "IP - INITIAL POINT"
42710  PRINT "IS - INITIAL STEP"
42720  PRINT "SD - REQ. S.D. FOR FN. CALLS"
42730  PRINT "MI - MAX. NO. OF FUNCTION CALLS"
42740  PRINT "CO - CONSOLE DISPLAY CODE"
42750  PRINT "PR - PRINTER DISPLAY CODE"
42800  PRINT "CC - ITERS. BET. CONVERGENCE CHECKS"
42810  PRINT "SR - STEP REDUCTION ON RESTART"
42820  PRINT "CD - LIST OF CODES"
42830  PRINT "NO - NO MORE CHANGES"
42840  RETURN
43000  REM  DATA TO FILE ****
43010  PRINT D0$;"OPEN"$N8$
43020  PRINT D0$;"DELETE"$N8$
43030  PRINT D0$;"OPEN"$N8$
43040  PRINT D0$;"WRITE"$N8$
43100  PRINT N
43110  FOR I9 = 1 TO N: PRINT X0(I9): NEXT
43120  FOR I9 = 1 TO N: PRINT S0(I9): NEXT
43200  PRINT R5: PRINT M0: PRINT C0: PRINT P0
43210  PRINT I5: PRINT S5
43300  PRINT D0$;"CLOSE"
43310  RETURN
43500  REM  LIST DATA ****
43505  REM  ASSUMES WO PAGE COLS. ON ENTRY
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..04
STARTING LINE..0    ENDING LINE..63999

43510  PRINT : PRINT N;" VARIABLE(S)"
43600  PRINT "INITIAL POINT:"
43610  FOR I9 = 1 TO N:V0(I9) = X0(I9): NEXT
43620  GOSUB 59500: REM  VECTOR LIST
43630  PRINT "INITIAL STEP:"
43640  FOR I9 = 1 TO N:V0(I9) = S0(I9): NEXT
43650  GOSUB 59500: REM  VECTOR LIST
43700  PRINT "REQUIRED STD. DEV. = "$R5
43710  PRINT "MAX. OF "$M0$" FUNCTION CALLS"
43720  PRINT "CONSOLE DISPLAY CODE = "$C0
43730  PRINT "PRINTER DISPLAY CODE = "$P0
43740  PRINT "CONV. CHK. EACH "$I5$" ITERATIONS"
43750  PRINT "STEP REDUCED BY "$S5$" ON RESTART"
43800  RETURN
44000  REM   PRINT DATA ****
44010  IF P0 <  = 0 THEN  RETURN
44020 W0 = 79
44030  PRINT D0$;"PR#"$S0%
44040  GOSUB 43500
44050  PRINT D0$;"PR#0"
44100  RETURN
44500  REM  RESULTS ON CONSOLE ****
44501  REM  (IF  NOT ALREADY DONE)
44510  IF C0 > 0 THEN  RETURN
44520 W0 = 39: REM  PAGE COLS.
44530  GOSUB 57700
44531  REM  LISTS FINAL DATA
44540  RETURN
50000  REM  SIMPLEX MIN. ALGORITHM **** $$$$$
50010 S0% = 1: REM  PRINTER SLOT, IF  USED
50020 D0$ =  CHR$ (4): REM  CTRL-D
50030  GOSUB 51000
50031  REM  INITIALIZES
50040 N1 = 0:I0 = 0:K0 =  - 1
50041  REM  CALLS, ITERS., RESTARTS
50100  REM  START, RESTART BLKL $$$$$
50110 K0 = K0 + 1
50120  GOSUB 51500
50121  REM  BUILDS INITIAL SIMPLEX
50130  GOSUB 52000
50131  REM  FINDS H1 AND L0
50140  IF K0 = 0 THEN  GOSUB 52500
50141  REM  PRINTS INIT. DATA
50200  REM  BEGIN ITERATION BLKL $$$$$
50210 I0 = I0 + 1
50220 H5 = 0:H6 = 0:H7 = 0:H8 = 0
50221  REM  CODES - REFLECT, EXTEND, CONTRACT, REDUCE
50300  GOSUB 53500
50301  REM  FINDS CENTROID
50310  GOSUB 54000
50311  REM  CONDUCTS SEARCH
```

```
NAML..MICRO SIMPLEX   DATL..03/13/81   TIML..1600   PAGL..05
STARTING LINL..0    ENDING LINL..63999

50320  GOSUB 52000
50321  REM  FINDS H1 AND L0
50330  GOSUB 53000
50331  REM  PRINTS CURRENT DATA
50340 I5 =  INT (15)
50350  IF 15 <  = 0 THEN 15 = 1
50360 E5 = 0
50400  IF 10 / 15 =  INT (10 / 15) THEN  GOSUB 56000
50401  REM  GET EXIT STATUS E5 (-1,0,1, OR 2)
50410  IF E5 = 0 GOTO 50200: REM  RE-ITER
50420  IF E5 < 2 GOTO 50500: REM  EXIT
50430  GOTO 50100: REM  RESTART
50500  REM  FINISH HERE $$$$$
50510  GOSUB 57000
50511  REM  SETS ALGORITHM OUTPUT
50520  GOSUB 57500
50521  REM  PRINTS FINAL DATA
50530  RETURN
51000  REM  INITIALIZE THE ALGORITHM $$$$
51010 R6 = 1:E6 = 2:C6 = .5:Q6 = .5
51011  REM  REFL., EXTEN., CONTR., REDUC. COEFFS.
51015  GOSUB 51020: GOTO 51200
51020 W0 = 0: ONERR  GOTO 59000
51030  DIM P5(N),P6(N),P7(N)
51031  REM  CENTROID, 1ST STEP, 2ND STEP
51040  DIM P9(N,N + 1),Y9(N + 1)
51041  REM  SIMPLEX, FUNCTION VALUES
51045  POKE 216,0: REM  RESET ERR FLAG
51100  RETURN
51200 N5 = N + 1:R7 = R5 * R5
51210 D5 = 1: REM  NO INIT. STEP REDUC.
51300  PRINT : PRINT "FREE REMAINING SPACE = ";
51310 W5 =  FRE (0)
51320 W6 = W5: IF W5 < 0 THEN W6 = 2 ↑ 16 + W5
51330  PRINT W6;
51340  PRINT " BYTES"
51400  RETURN
51500  REM  BUILD INIT. SIMPLEX $$$$
51510  IF R0 > 0 THEN  GOTO 51700
51520  FOR I9 = 1 TO N
51530 P9(I9,N5) = X0(I9): NEXT
51600 NF = NF + 1: GOSUB 1000: REM  CALL ####
51610 Y9(N5) = F
51700  FOR J9 = 1 TO N
51710  FOR I9 = 1 TO N
51720 X(I9) = P9(I9,N5)
51730  IF I9 = J9 THEN X(I9) = X(I9) + S0(I9) * D5
51740 P9(I9,J9) = X(I9)
51750  NEXT I9
51800 NF = NF + 1: GOSUB 1000: REM  CALL ####
51810 Y9(J9) = F
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..06
STARTING LINE...0    ENDING LINE..63999

51820  NEXT J9
51900  RETURN
52000  REM   FIND H1 AND LO $$$$
52010 Y6 = Y9(1):Y7 = Y6
52020 I6 = 1:I7 = I6
52100  FOR I9 = 1 TO N5
52110  IF Y9(I9) >  = Y6 THEN  GOTO 52200
52120 Y6 = Y9(I9):I6 = I9
52200  IF Y9(I9) <  = Y7 THEN  GOTO 52300
52210 Y7 = Y9(I9):I7 = I9
52250 W6 = Y9(I6)
52300  NEXT
52400  RETURN
52500  REM  INITIAL PRINT $$$$
52510 W0 = 39: REM  PAGE COLS.
52520  IF C0 > 0 THEN  GOSUB 52700
52530  IF P0 <  = 0 THEN  RETURN
52600 W0 = 79
52610  PRINT D0$;"PR#";S0%
52620  GOSUB 52700
52630  PRINT D0$;"PR#0"
52640  RETURN
52700  PRINT : PRINT "INITIAL POINT:"
52710  FOR I9 = 1 TO N
52720 V0(I9) = P9(I9,N5): NEXT
52750  GOSUB 59500
52800  PRINT "INITIAL FUNCTION:  ";Y9(N5)
52810  PRINT "INITIAL SIMPLEX H1 AND LO:"
52820  PRINT Y7;"    ";Y6
52830  RETURN
53000  REM  PRINT CURRENT RESULTS $$$$
53010 W0 = 39:W5 = C0
53020  IF C0 > 0 THEN  IF 10 / C0 =  INT (10 / C0) THEN  GOSUB 53
     200
53030  IF P0 > 0 THEN  IF 10 / P0 =  INT (10 / P0) THEN  GOSUB 53
     100
53040  RETURN
53100 W0 = 79:W5 = P0
53110  PRINT D0$;"PR#";S0%
53120  GOSUB 53200
53130  PRINT D0$;"PR#0"
53140  RETURN
53200  PRINT
53210  PRINT N7;" CALLS, ";I0;" ITERS., ";R0;" RESTARTS"
53220  IF W5 <  > 1 GOTO 53300
53225  PRINT "VALUE AT REFLECTED POINT IMPROVES"
53226  PRINT "  ";N9;" VALUE(S) AT SIMPLEX POINTS"
53230  PRINT "ACTION(S) ACCEPTED THIS ITERATION:"
53235  PRINT "  ";
53240  IF H5 = 1 THEN  PRINT "REFLECTION ";
53250  IF H6 = 1 THEN  PRINT "EXTENSION ";
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..07
STARTING LINE..0    ENDING LINE..63999

53260  IF H7 = 1 THEN  PRINT "CONTRACTION ";
53270  IF H8 = 1 THEN  PRINT "REDUCTION ";
53280  PRINT
53300  PRINT "CURRENT MINIMUM FOUND AT:"
53310  FOR I9 = 1 TO N
53320 V0(I9) = P9(I9,16): NEXT
53350  GOSUB 59500
53400  PRINT "CURRENT HI AND LO:"
53410  PRINT Y7;"    ";Y6
53420  RETURN
53500  REM  FIND CENTROID $$$$
53510  FOR I9 = 1 TO N
53520 W5 = 0
53530  FOR J9 = 1 TO N5
53540 W5 = W5 + P9(I9,J9): NEXT
53600 W5 = W5 - P9(I9,17)
53610 P5(I9) = W5 / N
53620  NEXT I9
53700  RETURN
54000  REM  CONDUCT SEARCH $$$$ #####
54010  GOSUB 55000: REM  REFLECT
54020  GOSUB 55500: REM  GET N9
54030  REM  N9 IS NO. OF Y'S WORSE THAN NEW Y
54100  IF N9 = 0 THEN  GOTO 54200
54101  REM  REFL. NO GOOD
54110  IF N9 = 1 THEN  GOTO 54300
54111  REM  REFL. JUST OK
54120  IF 1 < N9 THEN  IF  N9 < N5 THEN  GOTO 54400
54121  REM  REFL. GOOD
54130  IF N9 = N5 THEN  GOTO 54500
54131  REM  REFL. EXCELLENT
54150  PRINT "ERROR IN N9": STOP
54151  REM  PROGRAM SHOULDN'T REACH THIS POINT
54200  REM  CASE N9=0 #####
54210  GOSUB 55200: REM  CONTRACT
54220  IF Z7 > Y7 THEN  GOSUB 55300: RETURN
54221  REM   REDUCE AND EXIT SEARCH
54230 H7 = 1: REM  CONTRACT OK
54240  GOSUB 55700: RETURN
54241  REM   ACCEPT 2ND STEP AND EXIT SEARCH
54300  REM  CASE N9=1 #####
54310  GOSUB 55600: REM  ACCEPT 1ST STEP
54320 Y7 = Y9(I7)
54330  GOTO 54200
54400  REM   CASE 1<N9<N+1 #####
54410  GOSUB 55600: REM  ACCEPT 1ST STEP
54420  RETURN : REM  EXIT SEARCH
54500  REM   CASE N9=N+1 #####
54520  GOSUB 55100: REM  EXTEND
54530  IF Z7 >  = Z6 THEN  GOSUB 55600: RETURN
54531  REM   ACCEPT 1ST AND EXIT SEARCH
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..08
STARTING LINE..0    ENDING LINE..63999

54540 H6 = 1: REM  EXT. OK
54550  GOSUB 55700: REM  ACCEPT 2ND
54560  RETURN : REM  EXIT SEARCH
55000  REM  REFLECT ####
55010  FOR I9 = 1 TO N
55020 P6(I9) = (1 + R6) * P5(I9) - R6 * P9(I9,17)
55030 X(I9) = P6(I9)
55040  NEXT
55050 NF = NF + 1: GOSUB 1000: REM  CALL ****
55060 Z6 = F
55070  RETURN
55100  REM  EXTEND ####
55110  FOR I9 = 1 TO N
55120 P7(I9) = E6 * P6(I9) + (1 - E6) * P5(I9)
55130 X(I9) = P7(I9)
55140  NEXT
55150 NF = NF + 1: GOSUB 1000: REM  CALL ****
55160 Z7 = F
55170  RETURN
55200  REM  CONTRACT ####
55210  FOR I9 = 1 TO N
55220 P7(I9) = C6 * P9(I9,17) + (1 - C6) * P5(I9)
55230 X(I9) = P7(I9)
55240  NEXT
55250 NF = NF + 1: GOSUB 1000: REM  CALL ****
55260 Z7 = F
55270  RETURN
55300  REM  REDUCE ####
55310  FOR J9 = 1 TO N5
55320  IF J9 = 16 THEN  GOTO 55400
55330  FOR I9 = 1 TO N
55340 X(I9) = Q6 * P9(I9,J9) + (1 - Q6) * P9(I9,16)
55350 P9(I9,J9) = X(I9)
55360  NEXT I9
55370 NF = NF + 1: GOSUB 1000: REM  CALL ****
55380 Y9(J9) = F
55400  NEXT J9
55410 H8 = 1: REM  REDUCE OK
55420  RETURN
55500  REM   GET N9 ####
55510 N9 = 0
55520  FOR I9 = 1 TO N5
55530  IF Y9(I9) > Z6 THEN N9 = N9 + 1
55540  NEXT
55550  RETURN
55600  REM  ACCEPT 1ST STEP (REFL.) ####
55610 H5 = 1: REM  REFLECT OK
55620  FOR I9 = 1 TO N
55630 P9(I9,17) = P6(I9): NEXT
55650 Y9(17) = Z6
55660  RETURN
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..09
STARTING LINE..0    ENDING LINE..63999

55700  REM  ACCEPT 2ND STEP (EXT. OR CONTR.) ####
55710  FOR I9 = 1 TO N
55720 P9(I9,I7) = P7(I9): NEXT
55750 Y9(I7) = Z7
55760  RETURN
56000  REM  GET EXIT STATUS $$$$
56020  IF NF >  = M0 THEN E5 =  - 1: RETURN
56021  REM  TOO MANY CALLS; EXIT
56100 S7 = 0
56110  FOR I9 = 1 TO N5
56120 S7 = S7 + Y9(I9): NEXT
56140 S7 = S7 / N5
56150 S8 = 0
56160  FOR I9 = 1 TO N5
56170 W5 = Y9(I9) - S7:S8 = S8 + W5 * W5
56180  NEXT
56200  IF S8 / N >  = K7 THEN E5 = 0: RETURN
56201  REM   LARGE STD. DEV.; RE-ITERATE
56220  FOR I9 = 1 TO N
56230 X(I9) = P9(I9,16): NEXT
56300  FOR I9 = 1 TO N
56310 W5 = S0(I9) * S5
56320 X(I9) = X(I9) + W5
56330 NF = NF + 1: GOSUB 1000: REM  CALL ####
56340  IF F < Y6 GOTO 56600
56400 X(I9) = X(I9) - W5 - W5
56410 NF = NF + 1: GOSUB 1000: REM  CALL ####
56420  IF F < Y6 GOTO 56600
56430 X(I9) = X(I9) + W5
56450  NEXT I9
56500  REM  MIN. TEST PASSED
56510 E5 = 1: RETURN
56511  REM  PASSED TEST; EXIT
56600  REM  MIN. TEST FAILED
56610  FOR I9 = 1 TO N
56620 P9(I9,N5) = X(I9): NEXT
56650 Y9(N5) = F: REM  NEW LO
56700 D5 = S5: REM  STEP REDUCTION
56710 E5 = 2
56720  RETURN
56721  REM  REM FAILED TEST; RESTART
57000  REM  SET ALGORITHM OUTPUT $$$$
57010  FOR I9 = 1 TO N
57020 X(I9) = P9(I9,16): NEXT
57050 F = Y9(16)
57100 NF =  - NF
57110  RETURN
57500  REM  PRINT FINAL RESULTS $$$$
57510 W0 = 39
57520  IF C0 > 0 THEN  GOSUB 57700
57530  IF P0 <  = 0 THEN  RETURN
```

```
NAME..MICRO SIMPLEX   DATE..03/13/81   TIME..1600   PAGE..10
STARTING LINE..0    ENDING LINE..63999

57600 W0 = 79
57610  PRINT D0$;"PR#";S0%
57620  GOSUB 57700
57630  PRINT D0$;"PR#0"
57640  RETURN
57700  PRINT
57710  PRINT  - NF;" CALLS, ";I0;" ITERS., ";R0;" RESTARTS"
57720  IF E5 =  - 1 THEN  PRINT "MAX. NO. OF FUNCTION CALLS REACH
     ED"
57730  IF E5 = 1 THEN  PRINT "FINAL POINT PASSED MINIMUM TEST"
57800  PRINT "MINIMUM FOUND AT:"
57810  FOR I9 = 1 TO N:V0(I9) = X(I9): NEXT
57850  GOSUB 59500
57900  PRINT "FN. VALUE AT THIS POINT:  ";F
57910  PRINT
57920  RETURN
59000  REM   IGNORE DEC 9/80
59001  REM   SUBROUTINE TO IGNORE AN ERROR.
59002  REM   CALL WITH "ONERR GOTO 59000".
59003  REM   ACCOMPLISHES A RETURN TO THE POINT AT WHICH
59004  REM   THE SUBROUTINE WITH ERROR WAS CALLED.
59006  REM   IF W0=1, AN ERROR MESSAGE IS PRINTED.
59010  POKE 216,0: REM   RESET ERROR FLAG
59020  POKE 768,104: POKE 769,168
59030  POKE 770,104: POKE 771,166
59040  POKE 772,223: POKE 773,154
59050  POKE 774,72: POKE 775,152
59060  POKE 776,72: POKE 777,96
59100  CALL 768: REM   RESET STACK
59110  IF W0 <  > 1 THEN 59200
59120  PRINT "ERROR NO. "; PEEK (222);" IGNORED."
59200  RETURN
59500  REM  VECTOR LIST
59510  REM  VECTOR=V0(N), PAGE COLS.=W0
59600 W5 = 0
59610  FOR I9 = 1 TO N
59620 W6 =  LEN ( STR$ (V0(I9))) + 3
59630  IF W5 + W6 > W0 THEN  PRINT :W5 = 0
59640 W5 = W5 + W6
59650  PRINT V0(I9);"   ";
59660  NEXT
59700  PRINT
59710  RETURN
```

** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...MICRO SIMPLEX
DATE...03/13/81

A0$
220 230 240 250 250 440 450 530 540 610 620 630 640 640 42040 42100
42110 42120 42130 42140 42150 42160 42170 42200 42210

C0
40700 41320 42140 43200 43720 44510 52520 53010 53020 53020 53020 57520

C6
51010 55220 55220

D0$
160 40510 40520 40720 43010 43020 43030 43040 43300 44030 44050 50020
52610 52630 53110 53130 57610 57630

D5
51210 51730 56700

E5
50360 50410 50420 56020 56200 56510 56710 57720 57730

E6
51010 55120 55120

F
51610 51810 55060 55160 55260 55380 56340 56420 56650 57050 57900

H5
50220 53240 55610

H6
50220 53250 54540

H7
50220 53260 54230

H8
50220 53270 55410

I0
50040 50210 50210 50400 50400 53020 53020 53030 53030 53210 57710

I5
40710 41410 42160 43210 43740 50340 50340 50350 50350 50400 50400

I6
52020 52020 52120 52250 53320 55320 55340 56230 57020 57050

I7
52020 52210 53600 54320 55020 55220 55630 55650 55720 55750

I9
360 360 360 460 460 460 40620 40620 40630 40630 41110 41110 41130 41130
42510 42510 42610 42610 43110 43110 43120 43120 43610 43610 43610
43640 43640 43640 51520 51530 51530 51710 51720 51720 51730 51730
51730 51730 51740 51740 51750 52100 52110 52120 52120 52200 52210
52210 52710 52720 52720 53310 53320 53320 53510 53540 53600 53610
53620 55010 55020 55020 55020 55030 55030 55110 55120 55120 55120
55130 55130 55210 55220 55220 55220 55230 55230 55330 55340 55340
55340 55350 55350 55360 55520 55530 55620 55630 55630 55710 55720
55720 56110 56120 56160 56170 56220 56230 56230 56300 56310 56320
56320 56400 56400 56430 56430 56450 56610 56620 56620 57010 57020
57020 57810 57810 57810 59610 59620 59650

J9
51700 51730 51740 51810 51820 53530 53540 55310 55320 55340 55350 55380
55400

M0
40700 41240 42130 43200 43710 56020

N

```
360 460 40600 40610 40610 40610 40610 40620 40630 41020 41040 41040
41040 41040 41110 41130 42510 42610 43100 43110 43120 43510 43610
43640 51030 51030 51030 51040 51040 51040 51200 51520 51700 51710
52710 53310 53510 53610 55010 55110 55210 55330 55620 55710 56200
56220 56300 56610 57010 57810 59610

N5
51200 51530 51610 51720 52100 52720 52800 53530 54120 54130 55310 55520
56110 56140 56160 56620 56650

N8$
40020 40510 40520 43010 43020 43030 43040

N9
53226 54100 54110 54120 54120 54130 55510 55530 55530

NF
350 50040 51600 51600 51800 51800 53210 55050 55050 55150 55150 55250
55250 55370 55370 56020 56330 56330 56410 56410 57100 57100 57710

P0
40700 41350 42150 43200 43730 44010 52530 53030 53030 53030 53100 57530

P5(
51030 53610 55020 55120 55220

P6(
51030 55020 55030 55120 55630

P7(
51030 55120 55130 55220 55230 55720

P9(
51040 51530 51720 51740 52720 53320 53540 53600 55020 55220 55340 55340
55350 55630 55720 56230 56620 57020

Q6
51010 55340 55340

R0
50040 50110 50110 50140 51510 53210 57710

R5
40700 41220 42120 43200 43700 51200 51200

R6
51010 55020 55020

R7
51200 56200

S0%
170 44030 50010 52610 53110 57610

S0(
40610 40630 41040 41130 42610 43120 43640 51730 56310

S5
40710 41430 42170 43210 43750 56310 56700

S7
56100 56120 56120 56140 56140 56170

S8
56150 56170 56170 56200

V0(
40610 41040 43610 43640 52720 53320 57810 59620 59650

W0
505 40605 41030 44020 44520 51020 52510 52600 53010 53100 57510 57600
59110 59630

W5
51310 51320 51320 51320 53010 53100 53220 53520 53540 53540 53600 53600
```

53610 56170 56170 56170 56310 56320 56400 56400 56430 59600 59630
59630 59640 59640

W6
51320 51320 51330 52250 59620 59630 59640

X(
360 460 40610 41040 51720 51730 51730 51740 55030 55130 55230 55340
55350 56230 56320 56320 56400 56400 56430 56430 56620 57020 57810

X0(
360 460 40610 40620 41040 41110 42510 43110 43610 51530

Y6
52010 52010 52110 52120 52820 53410 56340 56420

Y7
52010 52200 52210 52820 53410 54220 54320

Y9(
51040 51610 51810 52010 52110 52120 52200 52210 52250 52800 54320 55380
55530 55650 55750 56120 56170 56650 57050

Z6
54530 55060 55530 55650

Z7
54220 54530 55160 55260 55750

```
]
NAME..MINV1    DATE..03/13/81    TIME..1600    PAGE..01
STARTING LINE..0     ENDING LINE..63999

38500  REM  MINV1 AUG 25/80
38501  REM  REV SEP 12/80
38505  REM  MATRIX INVERSE WITHOUT PIVOT
38510 A0 = 1:B0 = 0:DA = A0
38600  FOR A7 = A0 TO NA
38610 A9 = A(A7,A7):A(A7,A7) = A0
38615 DA = DA * A9: IF DA = 0 THEN  RETURN
38620  FOR A6 = A0 TO NA:A(A7,A6) = A(A7,A6) / A9: NEXT
38630  FOR A5 = A0 TO NA: IF  A5 = A7 THEN 38700
38640 A8 = A(A5,A7):A(A5,A7) = B0
38650  FOR A6 = A0 TO NA:A(A5,A6) = A(A5,A6) - A8 * A(A7,A6): NEXT

38700  NEXT : NEXT : RETURN
```

** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...M1NV1
DATE...03/13/81

A(
38610 38610 38620 38620 38640 38640 38650 38650 38650

A0
38510 38510 38600 38610 38620 38630 38650

A5
38630 38630 38640 38640 38650 38650

A6
38620 38620 38620 38650 38650 38650 38650

A7
38600 38610 38610 38610 38610 38620 38620 38630 38640 38640 38650

A8
38640 38650

A9
38610 38615 38620

B0
38510 38640

DA
38510 38615 38615 38615

NA
38600 38620 38630 38650

```
]
NAME..M1NV2   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

39000  REM  M1NV2 AUG 30/80
39001  REM  REV SEP 26/80
39005  REM  MATRIX INVERSE WITH PIVOT
39010 A0 = 1:B0 = 0:DA = A0
39020  FOR A7 = A0 TO NA:A9 = B0
39100  FOR A5 = A7 TO NA
39110  FOR A6 = A7 TO NA
39120  IF  ABS (A9) >  =  ABS (A(A5,A6)) THEN 39190
39130 A9 = A(A5,A6):A(A7,0) = A5:A(0,A7) = A6
39190  NEXT : NEXT
39200 DA = DA * A9: IF DA = 0 THEN  RETURN
39300 A5 = A(A7,0): IF A5 = A7 THEN 39400
39310 DA =  - DA
39320  FOR A6 = A0 TO NA
39330 A8 = A(A7,A6):A(A7,A6) = A(A5,A6):A(A5,A6) = A8: NEXT
39400 A6 = A(0,A7): IF A6 = A7 THEN 39500
39410 DA =  - DA
39420  FOR A5 = A0 TO NA
39430 A8 = A(A5,A7):A(A5,A7) = A(A5,A6):A(A5,A6) = A8: NEXT
39500 A(A7,A7) = A0
39510  FOR A6 = A0 TO NA:A(A7,A6) = A(A7,A6) / A9: NEXT
39550  FOR A5 = A0 TO NA: IF  A5 = A7 THEN 39650
39560 A8 = A(A5,A7):A(A5,A7) = B0
39600  FOR A6 = A0 TO NA:A(A5,A6) = A(A5,A6) - A8 * A(A7,A6): NEXT

39650  NEXT : NEXT
39800  FOR A7 = NA TO A0 STEP  - A0
39810 A6 = A(A7,0): IF  A6 = A7 THEN 39900
39820  FOR A5 = A0 TO NA
39830 A8 = A(A5,A7):A(A5,A7) = A(A5,A6):A(A5,A6) = A8: NEXT
39900 A5 = A(0,A7): IF A5 = A7 THEN 39990
39910  FOR A6 = A0 TO NA
39920 A8 = A(A7,A6):A(A7,A6) = A(A5,A6):A(A5,A6) = A8: NEXT
39990  NEXT : RETURN
```

** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...MINV2
DATE...03/13/81

A(
39120 39130 39130 39130 39300 39330 39330 39330 39330 39400 39430 39430
39430 39430 39500 39510 39510 39560 39560 39600 39600 39600 39810
39830 39830 39830 39830 39900 39920 39920 39920 39920

A0
39010 39010 39020 39320 39420 39500 39510 39550 39600 39800 39800 39820
39910

A5
39100 39120 39130 39130 39300 39300 39330 39330 39420 39430 39430 39430
39430 39550 39550 39560 39560 39600 39600 39820 39830 39830 39830
39830 39900 39900 39920 39920

A6
39110 39120 39130 39130 39320 39330 39330 39330 39330 39400 39400 39430
39430 39510 39510 39510 39600 39600 39600 39600 39810 39810 39830
39830 39910 39920 39920 39920 39920

A7
39020 39100 39110 39130 39130 39300 39300 39330 39330 39400 39400 39430
39430 39500 39500 39510 39510 39550 39560 39560 39600 39800 39810
39810 39830 39830 39900 39900 39920 39920

A8
39330 39330 39430 39430 39560 39600 39830 39830 39920 39920

A9
39020 39120 39130 39200 39510

B0
39010 39020 39560

DA
39010 39200 39200 39200 39310 39310 39410 39410

NA
39020 39100 39110 39320 39420 39510 39550 39600 39800 39820 39910

```
]
NAME..MULTIMIN   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1000  REM  MULTIMIN AUG 13/80
1001  REM  REV MAR 12/81
1010  REM   FUNCTION FOR MICRO SIMPLEX
1020  REM   DESCRIPTION PRINTED AUTOMATICALLY
1030  REM   SEE LINE 5000
1100  IF NF > 0 THEN  GOSUB 2000: RETURN
1101  REM   CALCULATES
1200  IF NF = 0 THEN  GOSUB 4000: RETURN
1201  REM   INITIALIZES
1300  IF NF < 0 THEN  GOSUB 6000: RETURN
1301  REM   FINALIZES
1400  RETURN
2000  REM   CALCULATE F *****
2010 Y = 1
2020 F = 0
2100  FOR I = 1 TO N
2110 Y = Y * X(I)
2120 F = F + A * X(I) * X(I) - B *  COS (P * Y)
2150  NEXT
2200  RETURN
4000  REM   INITIALIZE *****
4010 D$ =  CHR$ (4):SL = 1
4011  REM   CTRL-D, PRINTER SLOT (IF USED)
4030  GOSUB 5000: REM   HEADER
4040  PRINT
4100  INPUT "WHAT IS A? ";A
4110  INPUT "WHAT IS B? ";B
4120  IF A < 0 THEN A =  - A
4130  IF B < 0 THEN B =  - B
4140 P = 3.14159265
4200  PRINT : PRINT "COPY FUNCTION DESCRIPTION ON PRINTER?"
4203  INPUT "Y OR N? ";R$
4206  IF R$ <  > "Y" THEN  RETURN
4210  PRINT D$;"PR#";SL
4220  GOSUB 5000: REM   HEADER
4230  PRINT
4240  PRINT "IN THIS RUN:"
4250  PRINT "  A = ";A;",  B = ";B
4260  PRINT D$;"PR#0"
4300  RETURN
5000  REM   HEADER
5010  PRINT
5020  PRINT "THIS FUNCTION HAS THE FORM:"
5030  PRINT "F = A * (SUM OF  SQUARES OF  X'S)"
5040  PRINT "      - B * SUM OF  (COS(P1*Y(I)))"
5050  PRINT "WHERE A>=0, B>=0, AND Y(1) IS"
5060  PRINT "THE PRODUCT OF  THE FIRST I X'S."
5070  PRINT
5100  PRINT "FOR EXAMPLE, WHEN N=2:"
5110  PRINT "F = A*(X1↑2+X2↑2)"
5120  PRINT "      -B*COS(P1*X1)-B*COS(P1*X1*X2)"
```

```
NAME..MULTIMIN   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

5200  PRINT
5210  PRINT "THE MINIMUM OF F IS -B*N."
5220  PRINT "IF A=0 AND B>0, THIS IS ACHIEVED"
5230  PRINT "WHEN ALL Y'S ARE EVEN INTEGERS."
5240  PRINT "IF A>0 AND B>0, THE GLOBAL MIN (-B*N)"
5250  PRINT "OCCURS ONLY WHEN ALL X'S (AND Y'S)"
5260  PRINT "ARE ZERO; HOWEVER OTHER LOCAL MINIMA"
5270  PRINT "EXIST."
5300  RETURN
6000  REM   FINALIZE #####
6010 WO = 39: GOSUB 7000
6020  PRINT : PRINT "COPY FINAL OUTPUT ON PRINTER?"
6023  INPUT "Y OR N? ";R$
6026  IF R$ <  > "Y" THEN  RETURN
6030  PRINT D$;"PR#";SL
6040 WO = 79: GOSUB 7000
6050  PRINT D$;"PR#0"
6060  RETURN
7000  REM   FINAL PRINT
7010  PRINT
7020  PRINT "A = ";A;",  B = ";B
7030  PRINT "Y'S (PRODUCTS OF FIRST I X'S):"
7100 Y = 1
7110  FOR I = 1 TO N
7120 Y = Y * X(I):VO(I) = Y
7130  NEXT
7150  GOSUB 59500: REM   VECTOR LIST
7200  PRINT "GRADIENT AT FINAL POINT:"
7300  FOR I = 1 TO N
7310 G = 2 * A * X(I)
7320 Y1 = 1
7325  IF I = 1 THEN  GOTO 7350
7330  FOR J = 1 TO I - 1
7340 Y1 = Y1 * X(J): NEXT J
7350 Y = Y1
7400  FOR J = I TO N
7410 Y = Y * X(J)
7420  IF J > I THEN Y1 = Y1 * X(J)
7430 G = G + P * Y1 * B *  SIN (P * Y)
7440  NEXT J
7450 VO(I) = G
7460  NEXT I
7500  GOSUB 59500: REM   VECTOR LIST
7510  RETURN
```

```
   ** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...MULTIMIN
        DATE...03/13/81

A
2120 4100 4120 4120 4120 4250 7020 7310

B
2120 4110 4130 4130 4130 4250 7020 7430

D$
4010 4210 4260 6030 6050

F
2020 2120 2120

G
7310 7430 7430 7450

I
2100 2110 2120 2120 7110 7120 7120 7300 7310 7325 7330 7400 7420 7450
7460

J
7330 7340 7340 7400 7410 7420 7420 7440

N
2100 7110 7300 7400

NF
1100 1200 1300

P
2120 4140 7430 7430

R$
4203 4206 6023 6026

SL
4010 4210 6030

V0(
7120 7450

W0
6010 6040

X(
2110 2120 2120 7120 7310 7340 7410 7420

Y
2010 2110 2110 2120 7100 7120 7120 7120 7350 7410 7410 7430

Y1
7320 7340 7340 7350 7420 7420 7430
```

```
]
NAME..PREDICT   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  PREDICT DEC 5/80
2  REM  REV MAR 12/81
100 D$ =  CHR$ (4): REM  CTRL-D
110 G$ =  CHR$ (7): REM  BELL
120 S% = 1: REM  PRINTER SLOT (IF USED)
200  GOSUB 40000: REM  INITIALIZE
210  GOSUB 40800: REM  PREPARE SCREEN
220  GOSUB 41000: REM   GET OBSERVATIONS, MIN, MAX
230  GOSUB 42000: REM  SET X-RANGE
240 R9 = 0: REM  INITIAL RUN CODE
250  GOSUB 42500: REM  GET PARAMETERS, PREDICTIONS, MIN, MAX
260  IF R9 = 0 THEN  GOSUB 43500: REM  SET Y-RANGE
300  POKE  - 16304,0: REM  SELECT SCREEN
310  IF R9 = 0 THEN  GOSUB 44000: REM  PLOT OBSERVATIONS
320  GOSUB 44500: REM  PLOT PREDICTIONS
330  GOSUB 45000: REM  PRINT RANGES, BELL
400  GOSUB 45500: REM  SUMMARY ON PRINTER
420  GOSUB 46500: REM  GRAPH ON DISK
430  GOSUB 47000: REM  RESTART
500 R9 = 1: REM  SECONDARY RUN CODE
510  GOTO 250
40000  REM  INITIALIZE
40010  GOSUB 40100: REM  MESSAGES
40020  GOSUB 40400: REM  SELECT OPTIONS
40030  RETURN
40100  HOME : VTAB 2
40110  PRINT "PROGRAM TO PLOT OBSERVATIONS (X,Y)"
40120  PRINT "AND/OR A PREDICTING CURVE OR CURVES."
40130  PRINT "USER PROGRAM CAN BE A SPECIALIZED"
40140  PRINT "VERSION OF  A FUNCTION FOR MICRO-SIMPLEX."
40200  PRINT "A PREDICTING CURVE TYPICALLY DEPENDS"
40210  PRINT "ON N PARAMETERS WHICH MIGHT HAVE BEEN"
40220  PRINT "FOUND OPTIMALLY BY MICRO SIMPLEX."
40230  PRINT
40300  PRINT "WARNING!...THIS PROGRAM SHOULD BE"
40310  PRINT "LOADED WITH HI-RES GRAPHICS PAGE 1"
40320  PRINT "PROTECTED. USE THE PROGRAM 'PROTECT'"
40330  PRINT "TO ACHIEVE THIS."
40340  PRINT
40350  RETURN
40400  PRINT "PLOT OBSERVATIONS, PREDICTIONS,"
40410  PRINT "  OR BOTH?"
40420  INPUT "O, P, OR B? ";R5$
40430  IF R5$ = "O" THEN  RETURN
40440  IF R5$ = "P" THEN  RETURN
40450  IF R5$ = "B" THEN  RETURN
40460  GOTO 40400
40800  REM  PREPARE SCREEN
40810  HGR : HCOLOR= 3
40820 A9 = 279:B9 = 159:C9 = 3:D9 = 2 * C9
40830  HPLOT C9,C9 TO C9,B9 - C9 TO A9 - C9,B9 - C9 TO A9 - C9,C9
```

```
NAME..PREDICT   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

      TO C9,C9
40840  POKE  - 16303,0: REM  RESTORE TEXT
40850  RETURN
41000  REM  GET OBSERVATIONS
41010  IF R5$ = "P" THEN  RETURN
41020 NP =  - 2: GOSUB 1500: REM ****USER ROUTINE****
41021  REM  GETS NO, XO(NO), YO(NO)
41100 A5 = XO(1):B5 = A5
41110 C5 = YO(1):D5 = C5
41200  FOR I5 = 1 TO NO
41210  IF XO(I5) < A5 THEN A5 = XO(I5)
41220  IF XO(I5) > B5 THEN B5 = XO(I5)
41230  IF YO(I5) < C5 THEN C5 = YO(I5)
41240  IF YO(I5) > D5 THEN D5 = YO(I5)
41250  NEXT
41300  IF A5 < B5 THEN 41400
41310  PRINT "NO X-RANGE IN THE OBSERVATIONS."
41320  PRINT "ALL X'S = ";A5
41330  STOP
41400  IF C5 < D5 THEN  RETURN
41410  PRINT "NO Y-RANGE IN THE OBSERVATIONS."
41420  PRINT "ALL Y'S = ";C5
41430  STOP
41500  RETURN
42000  REM  SET X-RANGE
42005  PRINT
42010  IF R5$ = "P" THEN 42300
42020  PRINT "WHAT IS THE EXPANSION FACTOR"
42030  PRINT "  FOR THE X-RANGE?"
42040  PRINT "EXAMPLES:"
42050  PRINT "  1  ...SAME AS OBSERVED RANGE"
42060  PRINT "  1.1...10% LARGER THAN OBSERVED RANGE"
42070  PRINT "  -1 ...RANGE TO BE SPECIFIED"
42100  PRINT : INPUT E5: PRINT
42110  IF E5 < 0 THEN 42300
42120  IF E5 < 1 THEN E5 = 1
42200 W5 = (E5 - 1) * (B5 - A5) / 2
42210 A7 = A5 - W5:B7 = B5 + W5
42220  RETURN
42300  PRINT "ENTER THE X-RANGE:"
42310  INPUT "  LOW X?  ";A7
42320  INPUT "  HIGH X? ";B7
42330  PRINT
42340  IF A7 < B7 THEN  RETURN
42350  PRINT "ERROR": GOTO 42300
42500  REM  GET PREDICTIONS
42510  IF R5$ = "O" THEN  RETURN
42520  IF R9 = 1 THEN 42600
42525 NP =  - 1: GOSUB 1500: REM ****USER ROUTINE****
42526  REM  GETS AUXILLARY MODEL PARAMETERS
42530  PRINT "HOW MANY PARAMETERS IN THE"
```

```
NAME..PREDICT    DATE..03/13/81    TIME..1600    PAGE..03
STARTING LINE..0     ENDING LINE..63999

42540  PRINT "  PREDICITING FUNCTION?"
42550  INPUT N
42560  IF N <  = 0 THEN 42800
42570  DIM X(N),V0(N)
42600  PRINT : PRINT "ENTER THE PARAMETERS:"
42610  FOR I5 = 1 TO N
42620  INPUT X(15):V0(15) = X(15)
42630  NEXT
42700  PRINT : PRINT "HERE ARE THE VALUES ENTERED:"
42710 W0 = 39: GOSUB 59500: REM  VECTOR LIST
42720  INPUT "OK? (Y OR N) ?";R6$
42730  IF R6$ = "N" THEN 42600
42740  PRINT
42800  IF R9 = 1 THEN 42900
42810  PRINT "HOW MANY POINTS ON THE"
42820  PRINT "  PREDICTING CURVE?"
42830  INPUT N5
42840  IF N5 < 2 THEN N5 = 2
42850  DIM X5(N5),Y5(N5)
42900 W5 = (B7 - A7) / (N5 - 1)
42910  FOR 15 = 1 TO N5
42920 XP = A7 + (15 - 1) * W5
42930 NP = 1: GOSUB 1500: REM ****USER PROGRAM****
42931  REM  COMPUTES YP FROM XP
42940 X5(15) = XP
42950 Y5(15) = YP
42960  NEXT
43000 A6 = X5(1):B6 = A6
43010 C6 = Y5(1):D6 = C6
43100  FOR 15 = 2 TO N5
43110  IF X5(15) < A6 THEN A6 = X5(15)
43120  IF X5(15) > B6 THEN B6 = X5(15)
43130  IF Y5(15) < C6 THEN C6 = Y5(15)
43140  IF Y5(15) > D6 THEN D6 = Y5(15)
43150  NEXT
43160  RETURN
43500  REM  SET Y-RANGE
43510  IF R5$ = "D" THEN 43600
43520  GOSUB 43900: REM  ALLOW SPECIFIED RANGE
43530  IF R5$ = "P" THEN 43700
43540  IF R5$ = "B" THEN 43800
43600 C7 = C5:D7 = D5
43610  RETURN
43700 C7 = C6:D7 = D6
43710  RETURN
43800 C7 = C5:D7 = D5
43810  IF C6 < C5 THEN C7 = C6
43820  IF D6 > D5 THEN D7 = D6
43830  RETURN
43900  PRINT "Y-RANGE SPECIFIED OR AUTOMATIC?"
43910  INPUT "S OR A? ";R6$
```

```
NAME..PREDICT   DATE..03/13/81   TIME..1600   PAGE..04
STARTING LINE..0    ENDING LINE..63999

43920  IF R6$ <  > "S" THEN  RETURN
43930  PRINT "ENTER THE Y-RANGE:"
43940  INPUT "  LOW YY  ";C7
43950  INPUT "  HIGH YY ";D7
43960  PRINT
43970  IF C7 < D7 THEN  POP : RETURN
43980  PRINT "ERROR": GOTO 43930
44000  REM  PLOT OBSERVATIONS
44010  IF R5$ = "P" THEN  RETURN
44100  FOR I5 = 1 TO N0
44110 P5 = C9 + (A9 - D9) * (X0(I5) - A7) / (B7 - A7)
44120 Q5 = C9 + (B9 - D9) * (D7 - Y0(I5)) / (D7 - C7)
44125  GOSUB 44400
44130  HPLOT P5 - C9,Q5 TO P5 + C9,Q5
44140  HPLOT P5,Q5 - C9 TO P5,Q5 + C9
44150  NEXT
44200  RETURN
44400  REM  CONFINE POINTS TO RANGES
44410  IF P5 < C9 THEN P5 = C9
44420  IF P5 > A9 - C9 THEN P5 = A9 - C9
44430  IF Q5 < C9 THEN Q5 = C9
44440  IF Q5 > B9 - C9 THEN Q5 = B9 - C9
44450  RETURN
44500  REM  PLOT PREDICTIONS
44510  IF R5$ = "O" THEN  RETURN
44520  FOR I5 = 1 TO N5
44530 P5 = C9 + (A9 - D9) * (X5(I5) - A7) / (B7 - A7)
44540 Q5 = C9 + (B9 - D9) * (D7 - Y5(I5)) / (D7 - C7)
44550  GOSUB 44400
44600  IF I5 = 1 THEN  HPLOT P5,Q5
44610  IF I5 > 1 THEN  HPLOT  TO P5,Q5
44620  NEXT
44700  RETURN
45000  REM  PRINT AXIS RANGES, BELL
45010  PRINT G5$: REM  BELL
45020  HOME : VTAB 21
45030  PRINT "X: "$A7;" TO ";B7
45040  PRINT "Y: "$C7;" TO ";D7
45050  PRINT "PRESS ANY KEY TO CONTINUE..."
45060  GET R6$
45100  RETURN
45500  REM  PRINTER SUMMARY
45510  PRINT "DO YOU WANT A PRINTED DATA SUMMARY?"
45520  INPUT "Y OR N? ";R6$
45530  IF R6$ <  > "Y" THEN  RETURN
45550  PRINT D5$;"PR#";S5%
45560  PRINT
45600  IF R5$ = "P" THEN 45700
45610  PRINT "OBSERVATIONS:"
45620  PRINT "      X FROM ";A5;" TO ";B5
45630  PRINT "      Y FROM ";C5;" TO ";D5
```

```
NAME..PREDICT   DATE..03/13/81   TIME..1600   PAGE..05
STARTING LINE..0    ENDING LINE..63999

45640  PRINT
45700  IF R5$ = "0" THEN 45800
45710  PRINT "PREDICTIONS:"
45720  PRINT "      X FROM ";A6$;" TO ";B6
45730  PRINT "      Y FROM ";C6$;" TO ";D6
45740  IF N <  = 0 THEN 45790
45750  PRINT "      BASED ON THE PARAMETERS:"
45760  FOR I5 = 1 TO N
45770  PRINT "           ";X(I5)
45780  NEXT
45790  PRINT
45800  PRINT "GRAPH:"
45810  PRINT "      X FROM ";A7$;" TO ";B7
45820  PRINT "      Y FROM ";C7$;" TO ";D7
45830  PRINT
45900  PRINT D5$;"PR#0"
45910  RETURN
46500  REM  SAVE GRAPH ON DISK
46510  PRINT "SAVE THIS PICTURE ON DISK?"
46520  INPUT "Y OR N? ";R6$
46530  IF R6$ <  > "Y" THEN  RETURN
46600  INPUT "SAVE IN WHAT FILE?";R6$
46610 R6$ = "PRED." + R6$
46620  PRINT D5$;"BSAVE ";R6$;",A$2000,L$2000"
46630  PRINT "PICTURE SAVED AS ";R6$
46640  RETURN
47000  REM  RESTART
47005  POKE  - 16303,0: REM  RESTORE TEXT
47010  IF R5$ = "0" THEN  STOP
47020  IF N <  = 0 THEN  STOP
47030  PRINT "PLOT ANOTHER PREDICTION WITH"
47040  PRINT "  DIFFERENT PARAMETERS?"
47050  INPUT "Y OR N? ";R6$
47100  IF R6$ = "N" THEN  STOP
47200  RETURN
59500  REM      VECTOR LIST
59510  REM    VECTOR-V0(N), PAGE COLS.=W0
59600 W5 = 0
59610  FOR I9 = 1 TO N
59620 W6 =  LEN ( STR$ (V0(I9))) + 3
59630  IF W5 + W6 > W0 THEN  PRINT :W5 = 0
59640 W5 = W5 + W6
59650  PRINT V0(I9);"    ";
59660  NEXT
59700  PRINT
59710  RETURN
```

```
   ** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...PREDICT
        DATE...03/13/81

A5
41100 41100 41210 41210 41300 41320 42200 42210 45620

A6
43000 43000 43110 43110 45720

A7
42210 42310 42340 42900 42920 44110 44110 44530 44530 45030 45810

A9
40820 40830 40830 44110 44420 44420 44530

B5
41100 41220 41220 41300 42200 42210 45620

B6
43000 43120 43120 45720

B7
42210 42320 42340 42900 44110 44530 45030 45810

B9
40820 40830 40830 44120 44440 44440 44540

C5
41110 41110 41230 41230 41400 41420 43600 43800 43810 45630

C6
43010 43010 43130 43130 43700 43810 43810 45730

C7
43600 43700 43800 43810 43940 43970 44120 44540 45040 45820

C9
40820 40820 40830 40830 40830 40830 40830 40830 40830 40830 40830 40830
44110 44120 44130 44130 44140 44140 44410 44410 44420 44420 44430
44430 44440 44440 44530 44540

D5
41110 41240 41240 41400 43600 43800 43820 45630

D5$
100 45550 45900 46620

D6
43010 43140 43140 43700 43820 43820 45730

D7
43600 43700 43800 43820 43950 43970 44120 44120 44540 44540 45040 45820


D9
40820 44110 44120 44530 44540

E5
42100 42110 42120 42120 42200

G5$
110 45010

I5
41200 41210 41210 41220 41220 41230 41230 41240 41240 42610 42620 42620
42620 42910 42920 42940 42950 43100 43110 43110 43120 43120 43130
43130 43140 43140 44100 44110 44120 44520 44530 44540 44600 44610
45760 45770

I9
59610 59620 59650

N
42550 42560 42570 42570 42610 45740 45760 47020 59610
```

```
N5
42830 42840 42840 42850 42850 42900 42910 43100 44520

N0
41200 44100

NP
41020 42525 42930

P5
44110 44130 44130 44140 44140 44410 44410 44420 44420 44530 44600 44610


Q5
44120 44130 44130 44140 44140 44430 44430 44440 44440 44540 44600 44610


R5$
40420 40430 40440 40450 41010 42010 42510 43510 43530 43540 44010 44510
45600 45700 47010

R6$
42720 42730 43910 43920 45060 45520 45530 46520 46530 46600 46610 46610
46620 46630 47050 47100

R9
240 260 310 500 42520 42800

S5%
120 45550

V0(
42570 42620 59620 59650

W0
42710 59630

W5
42200 42210 42210 42900 42920 59600 59630 59630 59640 59640

W6
59620 59630 59640

X(
42570 42620 42620 45770

X5(
42850 42940 43000 43110 43110 43120 43120 44530

XD(
41100 41210 41210 41220 41220 44110

XP
42920 42940

Y5(
42850 42950 43010 43130 43130 43140 43140 44540

YD(
41110 41230 41230 41240 41240 44120

YP
42950
```

```
]
NAME..PROFILE   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  PROFILE NOV 25/80
2  REM  REV FEB 26/81
100  HOME : VTAB 2
110 D5$ =  CHR$ (4): REM  CTRL-D
115 S5 = 1: REM  PRINTER SLOT
120  GOSUB 40000: REM  GET ALL PROFILE DESCRIPTIONS
130  GOSUB 41000: REM  CALCULATE PROFILES
140  GOSUB 42000: REM  GET ALL MIN'S AND MAX'S
150  GOSUB 43000: REM  PROFILE DATA TO PRINTER
200  HOME : VTAB 21
210  GOSUB 44000: REM  CLEAR SCREEN, DRAW BORDER
220  GOSUB 44500: REM  GET COMMAND
300  IF R0$ = "P" THEN  GOSUB 45000: REM  PLOT PROFILE
310  IF R0$ = "E" THEN 210: REM  ERASE
320  IF R0$ = "S" THEN  GOSUB 46000: REM  SAVE ON FILE
330  IF R0$ = "L" THEN  GOSUB 46500: REM  LOAD FILE
350  IF R0$ = "Q" THEN 600: REM  QUIT
400  GOTO 220
600  TEXT
610  END
40000  REM  SET UP PROFILE END POINTS
40010  PRINT "PROGRAM TO PLOT 'PROFILES' OF  A"
40020  PRINT "USER-DEFINED FUNCTION OF  N VARIABLES."
40030  PRINT "THE FUNCTION SHOULD BE CODED AS"
40040  PRINT "REQUIRED FOR MICRO SIMPLEX."
40050  PRINT
40100  PRINT "A PROFILE IS A GRAPH OF  FUNCTION"
40110  PRINT "VALUES ALONG A LINE BETWEEN TWO"
40120  PRINT "SPECIFIED POINTS IN N-SPACE."
40130  PRINT
40140  PRINT "WARNING!...THIS PROGRAM SHOULD BE"
40150  PRINT "LOADED WITH HI-RES GRAPHICS PAGE 1"
40160  PRINT "PROTECTED. USE PROGRAM 'PROTECT'"
40170  PRINT "TO ACHIEVE THIS."
40180  PRINT
40200  INPUT "WHAT IS N? ";N
40210  INPUT "HOW MANY PROFILES IN THIS RUN? ";M0
40220  DIM X(N),P9(N,M0),V0(N)
40230  PRINT "HOW MANY POINTS ON EACH PROFILE?"
40240  PRINT "(MIN 2, MAX 280, SUGGEST 20 OR MORE)"
40250  INPUT K0
40260 K0 =  INT (K0)
40270  IF K0 <  = 2 THEN K0 = 2: IF K0 >  = 280 THEN K0 = 280
40300  PRINT "ENTER COORDINTES OF  THE MID-POINT"
40305  PRINT "  FOR ALL PROFILES:"
40310  FOR I5 = 1 TO N
40320  INPUT P9(I5,0):V0(I5) = P9(I5,0)
40340  NEXT
40350  GOSUB 40400
40360  IF R6$ = "N" THEN 40300
40370  GOTO 40500
```

```
NAME..PROFILE   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

40400  REM  POINT CONFIRMATION
40405 W0 = 39: REM  PAGE COLS.
40410  PRINT "HERE ARE THE COORDINATES JUST ENTERED:"
40420  GOSUB 59500: REM  VECTOR LIST
40430  INPUT "OK? (Y OR N) ";R6$
40440  RETURN
40500  FOR J5 = 1 TO M0
40510  PRINT "ENTER THE COORDINATES OF THE STEP TO"
40520  PRINT "  THE FINAL POINT FOR PROFILE ";J5;":"
40530  FOR I5 = 1 TO N
40540  INPUT P9(I5,J5):V0(I5) = P9(I5,J5)
40550  NEXT
40600  GOSUB 40400
40610  IF R6$ = "N" THEN 40510
40620  NEXT J5
40900  RETURN
41000  REM  CALCULATE PROFILES
41030  FOR I5 = 1 TO N:X(I5) = P9(I5,0): NEXT
41040 NF = 0
41050  GOSUB 1000: REM  INITIALIZE, IF NECESSARY
41060  PRINT : PRINT "PROFILE CALCULATIONS BEGIN. THIS"
41070  PRINT "MAY TAKE A WHILE..."
41100 NF = 1
41120  DIM F9(K0,M0)
41200  FOR J6 = 1 TO M0: REM  PROFILE NO.
41210  FOR I6 = 1 TO K0: REM  POINT NO.
41220 W5 = (I6 - 1) / (K0 - 1)
41230  FOR I5 = 1 TO N: REM  COORDINATE NO.
41240 X(I5) = P9(I5,0) + (2 * W5 - 1) * P9(I5,J6)
41250  NEXT
41300  GOSUB 1000
41310 F9(I6,J6) = F
41320  NEXT : NEXT
41400  RETURN
42000  REM  GET MINIMA AND MAXIMA
42010  DIM F5(M0),F6(M0)
42020  FOR J6 = 1 TO M0
42030 F5(J6) = F9(1,J6)
42040 F6(J6) = F9(1,J6)
42100  FOR I6 = 2 TO K0
42110  IF F9(I6,J6) < F5(J6) THEN F5(J6) = F9(I6,J6)
42120  IF F9(I6,J6) > F6(J6) THEN F6(J6) = F9(I6,J6)
42130  NEXT : NEXT
42200 G5 = F5(1):G6 = F6(1)
42210  IF M0 = 1 THEN 42300
42220  FOR I6 = 2 TO M0
42230  IF F5(I6) < G5 THEN G5 = F5(I6)
42240  IF F6(I6) > G6 THEN G6 = F6(I6)
42250  NEXT
42300  RETURN
43000  REM  PRINT PROFILE DATA
```

```
NAME..PROFILE   DATE..03/13/81   TIME..1600   PAGE..03
STARTING LINE..0    ENDING LINE..63999

43010  PRINT : PRINT  CHR$ (7): REM  BELL
43020  PRINT "PROFILE CALCULATIONS COMPLETE."
43030  PRINT "DO YOU WANT A SUMMARY OF PROFILE"
43040  PRINT "  DATA ON THE PRINTER?"
43100  INPUT "Y OR N? ";R6$
43110  IF R6$ < > "Y" THEN  RETURN
43120  PRINT "PREPARE PRINTER, 'RETURN' TO START..."
43200  GET R6$: IF R6$ < > CHR$ (13) THEN 43200
43205  PRINT
43210  PRINT D5$;"PR#";S5: PRINT
43220 W0 = 80: REM  PAGE COLS.
43300  PRINT "MID-POINT FOR ALL PROFILES:"
43310  FOR I5 = 1 TO N:V0(I5) = P9(I5,0): NEXT
43320  GOSUB 59500: REM  VECTOR LIST
43400  FOR J5 = 1 TO M0
43410  PRINT : PRINT "PROFILE ";J5;" BEGINS AT:"
43420  FOR I5 = 1 TO N:V0(I5) = P9(I5,0) - P9(I5,J5): NEXT
43430  GOSUB 59500: REM   VECTOR LIST
43440  PRINT "AND ENDS AT:"
43450  FOR I5 = 1 TO N:V0(I5) = P9(I5,0) + P9(I5,J5): NEXT
43460  GOSUB 59500: REM  VECTOR LIST
43470  NEXT
43500  PRINT
43510  PRINT "PROFILE","MINIMUM","MAXIMUM"
43520  FOR J5 = 1 TO M0
43530  PRINT "#";J5,F5(J5),F6(J5)
43540  NEXT
43550  PRINT "ALL",G5,G6
43600  PRINT : PRINT
43610  PRINT D5$;"PR#0"
43900  RETURN
44000  REM  SET UP SCREEN
44010  HGR : HCOLOR= 3
44020 A5 = 279:B5 = 159
44030  HPLOT 0,0 TO 0,B5 TO A5,B5 TO A5,0 TO 0,0
44400  RETURN
44500  REM  GET COMMAND
44510  PRINT
44520  PRINT "P)LOT, E)RASE, S)AVE, L)OAD,"
44530  PRINT " OR Q)UIT";
44600  GET R0$: PRINT
44900  RETURN
45000  REM  PLOT PROFILE
45010  PRINT
45020  INPUT "PLOT PROFILE NO. ";J6
45100  PRINT "PRO.";: HTAB 7: PRINT "MIN.";: HTAB 24: PRINT "MAX.
     "
45110  PRINT "#";J6;
45120  HTAB 7: PRINT F5(J6);
45130  HTAB 24: PRINT F6(J6)
45140  PRINT "ALL";: HTAB 7: PRINT G5;: HTAB 24: PRINT G6
```

```
NAME..PROFILE   DATE..03/13/81   TIME..1600   PAGE..04
STARTING LINE..0    ENDING LINE..63999

45200 A6 = A5 / (K0 - 1)
45210 B6 = B5 / (G6 - G5)
45220  HPLOT 0,0
45300  FOR I6 = 1 TO K0
45310 I5 = (I6 - 1) * A6
45320 J5 = B5 - (F9(I6,J6) - G5) * B6
45330  HPLOT  TO I5,J5
45340  NEXT
45400  PRINT "PRESS ANY KEY TO CONTINUE...";
45410  GET R6$: PRINT
45900  RETURN
46000  REM  SAVE ON FILE
46010  PRINT
46020  INPUT "SAVE IN WHAT FILE? ";R6$
46030 R6$ = "PRO." + R6$
46040  PRINT D5$;"BSAVE";R6$;",A$2000,L$2000"
46100  PRINT "CURRENT GRAPH SAVED AS ";R6$
46110  RETURN
46500  REM  LOAD FILE
46510  PRINT
46520  INPUT "LOAD WHAT FILE? ";R6$
46530 R6$ = "PRO." + R6$
46540  ONERR  GOTO 46700
46550  PRINT D5$;"BLOAD";R6$
46600  POKE 216,0: REM  RESET ERR FLAG
46610  PRINT R6$;" LOADED AND DISPLAYED."
46620  RETURN
46700  POKE 216,0: REM  RESET ERR FLAG
46710  POKE 768,104: POKE 769,169
46720  POKE 770,104: POKE 771,166
46730  POKE 772,223: POKE 773,154
46740  POKE 774,72: POKE 775,152
46750  POKE 776,72: POKE 777,96
46760  CALL 768: REM  RESET STACK
46800 W5 =  PEEK (222): REM  ERR CODE
46810 R8$ = "FILE " + R6$ + " NOT FOUND"
46820 R9$ = "FILE TYPE MISMATCH"
46830  IF W5 = 6 THEN  PRINT R8$: RETURN
46840  IF W5 = 13 THEN  PRINT R9$: RETURN
46900  PRINT "ERROR NO. ";W5
46910  PRINT "? FILE ERROR 6 OR 13 EXPECTED."
46920  STOP
59500  REM    VECTOR LIST
59510  REM   VECTOR=V0(N), PAGE COLS.=W0
59600 W5 = 0
59610  FOR I9 = 1 TO N
59620 W6 =  LEN ( STR$ (V0(I9))) + 3
59630  IF W5 + W6 > W0 THEN  PRINT :W5 = 0
59640 W5 = W5 + W6
59650  PRINT V0(I9);"   ";
59660  NEXT
```

```
NAME..PROFILE   DATE..03/13/81   TIME..1600   PAGE..05
STARTING LINE..0    ENDING LINE..63999

59700  PRINT
59710  RETURN
```

```
   ** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...PROJ1LL
        DATE...03/13/81

A5
44020 44030 44030 45200

A6
45200 45310

B5
44020 44030 44030 45210 45320

B6
45210 45320

D5$
110 43210 43610 46040 46550

F
41310

F5(
42010 42030 42110 42110 42200 42230 42230 43530 45120

F6(
42010 42040 42120 42120 42200 42240 42240 43530 45130

F9(
41120 41310 42030 42040 42110 42110 42120 42120 45320

G5
42200 42230 42230 43550 45140 45210 45320

G6
42200 42240 42240 43550 45140 45210

I5
40310 40320 40320 40320 40530 40540 40540 40540 41030 41030 41030 41230
41240 41240 41240 43310 43310 43310 43420 43420 43420 43420 43450
43450 43450 43450 45310 45330

I6
41210 41220 41310 42100 42110 42110 42120 42120 42220 42230 42230 42240
42240 45300 45310 45320

I9
59610 59620 59650

J5
40500 40520 40540 40540 40620 43400 43410 43420 43450 43520 43530 43530
43530 45320 45330

J6
41200 41240 41310 42020 42030 42030 42040 42040 42110 42110 42110 42110
42120 42120 42120 42120 45020 45110 45120 45130 45320

K0
40250 40260 40260 40270 40270 40270 40270 41120 41210 41220 42100 45200
45300

M0
40210 40220 40500 41120 41200 42010 42010 42020 42210 42220 43400 43520

N
40200 40220 40220 40220 40310 40530 41030 41230 43310 43420 43450 59610

NF
41040 41100
```

```
P9(
40220 40320 40320 40540 40540 41030 41240 41240 43310 43420 43420 43450
43450

R0$
300 310 320 330 350 44600

R6$
40360 40430 40610 43100 43110 43200 43200 45410 46020 46030 46030 46040
46100 46520 46530 46530 46550 46610 46810

R8$
46810 46830

R9$
46820 46840

S5
115 43210

V0(
40220 40320 40540 43310 43420 43450 59620 59650

W0
40405 43220 59630

W5
41220 41240 46800 46830 46840 46900 59600 59630 59630 59640 59640

W6
59620 59630 59640

X(
40220 41030 41240
```

```
]
NAME..PROTECT   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  PROTECT OCT 16/80
2  REM  REV OCT 17/80
100  HOME
105  VTAB 2
110  PRINT "PROGRAM TO PROTECT (OR DE-PROTECT)"
120  PRINT "GRAPHICS PAGES BY CONTROLLING"
130  PRINT "WHERE BASIC PROGRAMS RESIDE IN"
140  PRINT "THE APPLE'S MEMORY."
200  PRINT
210  PRINT "DO YOU WANT GRAPHICS PROTECTION?"
220  INPUT "Y OR N? ";R$
230  IF R$ = "Y" THEN 400
240  IF R$ <  > "N" THEN 220
300  PRINT
310  PRINT "NO PROTECTION: PROGRAMS BEGIN"
320  PRINT "ON MEMORY PAGE 8."
330  POKE 104,8: POKE 103,1
340  POKE 2048,0
350  GOTO 900
400  PRINT
410  PRINT "PROTECT WHICH PAGE?"
420  INPUT "1 OR 2? ";R$
430  IF R$ = "2" THEN 600
440  IF R$ <  > "1" THEN 420
500  PRINT
510  PRINT "GRAPHICS PAGE 1 PROTECTED: PROGRAMS"
520  PRINT "BEGIN ON MEMORY PAGE 64 ($40)."
530  POKE 104,64: POKE 103,1
540  POKE 16384,0
550  GOTO 900
600  PRINT
610  PRINT "GRAPHICS PAGE 2 PROTECTED: PROGRAMS"
620  PRINT "BEGIN ON MEMORY PAGE 96 ($60)."
630  POKE 104,96: POKE 103,1
640  POKE 24576,0
900  PRINT  CHR$ (4);"LOAD PROTECT"
```

```
]
NAME..REGRESS   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  REGRESS FEB 18/81
2  REM  REV MAR 12/81
1000  REM  **** SIMPLEX ENTRY POINT ****
1100  IF NF > 0 THEN 2000: REM  CALCULATE F
1110  IF NF = 0 THEN 4000: REM  INITIALIZE
1120  IF NF < 0 THEN 7000: REM  FINALIZE
1130  RETURN
1500  REM  $$$$ PREDICT ENTRY POINT $$$$
1600  IF NP = 1 THEN 3000: REM  PREDICT YP FROM XP
1610  IF NP =  - 1 THEN 5000: REM  SET UP MODEL PARAMETERS
1620  IF NP =  - 2 THEN 6000: REM  GET OBSERVED DATA
1630  RETURN
2000  REM  *** FUNCTION CALCULATION ***
2500  REM  SUM OF SQUARES
2510 F = 0
2520  IF E$ = "A" THEN 2600
2530  IF E$ = "L" THEN 2700
2540  RETURN
2600  FOR I = 1 TO NO
2610 XP = XO(I): GOSUB 3000
2620 ER = YO(I) - YP:F = F + ER * ER
2630  NEXT
2640  RETURN
2700  FOR I = 1 TO NO
2710 XP = XO(I): GOSUB 3000
2720 ER =  LOG (YO(I) / YP):F = F + ER * ER
2730  NEXT
2740  RETURN
3000  REM  $$$ PREDICT YP FROM XP $$$
3100 Y1 = X(1):A1 = X(2):A2 = X(3):B = X(4)
3110  IF B <  > 0 THEN XB = ((XP / X1) ↑ B - 1) / B
3120  IF B = 0 THEN XB =  LOG (XP / X1)
3130 YP = Y1 + A1 * XB + A2 * XB * XB
3999  RETURN
4000  REM  *** INITIALIZE ***
4800  GOSUB 5000: REM  SET UP MODEL PARAMETERS
4810  GOSUB 4900: REM  CHOOSE ERROR STRUCTURE
4820  GOSUB 6000: REM  GET OBSERVED DATA
4830  RETURN
4900  PRINT : PRINT "ERRORS ADDITIVE OR LOGARITHMIC?"
4910  INPUT "A OR L? ";E$
4920  IF  NOT (E$ = "A" OR E$ = "L") THEN 4900
4930  RETURN
5000  REM  $$$ SET UP MODEL PARAMETERS $$$
5100  PRINT : PRINT "REGRESSION OF  Y ON X↑B AND (X↑B)↑2:"
5110  PRINT
5120  PRINT "    Y = Y1 + A1*XB + A2*XB↑2"
5130  PRINT "    WHERE XB = ((X/X1)↑B-1)/B IF B<>0"
5140  PRINT "           XB = LOG(X/X1)      IF B=0"
5150  PRINT
5160  INPUT "WHAT IS X1? ";X1
```

```
NAME..REGRESS   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

5999  RETURN
6000  REM  $$$ GET OBSERVATIONS $$$
6010 ER$ = "N": GOSUB 6100: IF ER$ = "Y" THEN 6010: REM  GET DATA
     , REPEATING UNTIL NO ERROR
6020  GOSUB 6500: REM  LIST DATA
6030  RETURN
6100  PRINT : PRINT "NAME OF  OBSERVED DATA FILE?"
6110  INPUT F1$
6120 D$ =  CHR$ (4)
6130  ONERR  GOTO 6800
6140  PRINT D$;"LOCK";F1$: REM  CHECK IF FILE EXISTS, IF NOT, BRA
     NCH TO ERROR ROUTINE.
6150  PRINT D$;"OPEN";F1$
6160  PRINT D$;"READ";F1$
6200  INPUT IT,N1$,N2$,NO,BY
6210  REM  ITEMS/REC.(2), ITEM NAMES, NO. OF  RECS., AV. BYTES/REC
     .
6220  IF RN = 0 THEN  DIM XO(NO),YO(NO)
6230 RN = 1: REM  RUN CODE; DIMENSION STATEMENT BYPASSED IN LATER
      RUNS
6240  FOR I = 1 TO NO
6250  INPUT XO(I),YO(I)
6260  NEXT
6300  PRINT D$;"CLOSE";F1$
6310  POKE 216,0: REM  RESTORE ERROR FLAG
6320  RETURN
6500  PRINT : PRINT NO;" OBSERVATIONS READ FROM THE"
6510  PRINT "  TEXT FILE ";F1$;" ."
6520  PRINT "DO YOU WANT THEM LISTED ON THE"
6530  PRINT "  SCREEN, PRINTER, OR NEITHER?"
6540  INPUT "S, P OR N? ";RE$
6550  IF  NOT (RE$ = "S" OR RE$ = "P") THEN  RETURN
6560 SL = 1: REM  PRINTER SLOT
6570  IF RE$ = "P" THEN  PRINT D$;"PR#";SL
6600  PRINT : PRINT N1$,N2$
6610  FOR I = 1 TO NO
6620  PRINT XO(I),YO(I)
6630  NEXT
6640  PRINT D$;"PR#0"
6650  RETURN
6800 ER$ = "Y": REM  DEAL WITH ERROR
6810  POKE 216,0: REM  RESET ERROR FLAG
6820  POKE 768,104: POKE 769,168
6830  POKE 770,104: POKE 771,166
6840  POKE 772,223: POKE 773,154
6850  POKE 774,72: POKE 775,152
6860  POKE 776,72: POKE 777,96
6870  CALL 768: REM  RESET STACK
6900 CD =  PEEK (222): REM  ERROR CODE
6910  IF CD = 5 THEN  PRINT "ERROR: NOT ENOUGH DATA IN FILE"
6920  IF CD = 6 THEN  PRINT "ERROR: FILE NOT FOUND"
```

```
NAME..REGRESS   DATE..03/13/81   TIME..1600   PAGE..03
STARTING LINE..0    ENDING LINE..63999

6930  IF CD = 13 THEN  PRINT "ERROR: NOT A TEXT FILE"
6940  IF (CD = 5 OR CD = 6 OR CD = 13) THEN  RETURN
6950  PRINT "UNEXPECTED ERROR NO. ";CD
6960  STOP
7000  REM  *** FINAL OUTPUT ***
7100  PRINT : PRINT "HOW MANY FREE PARAMETERS IN THIS RUN?"
7110  INPUT FRZ
7120  IF (FRZ > 4 OR FRZ > NO - 1) THEN  PRINT "TOO MANY. RE-ENTE
     R.": GOTO 7100
7130 S1$ = "STD.":S3$ = "ERR."
7140 SE =  SQR (F / (NO - FRZ))
7150  IF E$ = "A" THEN S2$ = " "
7160  IF E$ = "L" THEN S2$ = " % ":SE = 100 * ( EXP (SE) - 1)
7200  PRINT : PRINT "X1 (FIXED)",X1
7210  PRINT "Y1",Y1
7220  PRINT "A1",A1
7230  PRINT "A2",A2
7240  PRINT "B",B
7250  PRINT S1$;S2$;S3$,SE
7999  RETURN
```

```
   ** THE VARIABLE CROSS REFERENCE **
PROGRAM NAME...REGRESS
        DATE...03/13/81

A1
3100 3130 7220

A2
3100 3130 7230

B
3100 3110 3110 3110 3120 7240

BY
6200

CD
6900 6910 6920 6930 6940 6940 6940 6950

D$
6120 6140 6150 6160 6300 6570 6640

E$
2520 2530 4910 4920 4920 7150 7160

ER
2620 2620 2620 2720 2720 2720

ER$
6010 6010 6800

F
2510 2620 2620 2720 2720 7140

FI$
6110 6140 6150 6160 6300 6510

FRX
7110 7120 7120 7140

I
2600 2610 2620 2700 2710 2720 6240 6250 6250 6610 6620 6620

IT
6200

N1$
6200 6600

N2$
6200 6600

NF
1100 1110 1120

NO
2600 2700 6200 6220 6220 6240 6500 6610 7120 7140

NP
1600 1610 1620

RE$
6540 6550 6550 6570

RN
6220 6230

S1$
7130 7250

S2$
7150 7160 7250

S3$
7130 7250

SE
7140 7160 7160 7250
```

```
SL
6560 6570

X(
3100 3100 3100 3100

X1
3110 3120 5160 7200

XB
3110 3120 3130 3130 3130

XO(
2610 2710 6220 6250 6620

XP
2610 2710 3110 3120

Y1
3100 3130 7210

YO(
2620 2720 6220 6250 6620

YP
2620 2720 3130
```

```
]
NAME..TEMPLATE   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  TEMPLATE MAR 10/81
2  REM  REV MAR 12/81
1000  REM  **** SIMPLEX ENTRY POINT ****
1100  IF NF > 0 THEN 2000: REM  CALCULATE F
1110  IF NF = 0 THEN 4000: REM  INITIALIZE
1120  IF NF < 0 THEN 7000: REM  FINALIZE
1130  RETURN
1500  REM  $$$$ PREDICT ENTRY POINT $$$$
1600  IF NP = 1 THEN 3000: REM  PREDICT YP FROM XP
1610  IF NP =  - 1 THEN 5000: REM  SET UP MODEL PARAMETERS
1620  IF NP =  - 2 THEN 6000: REM  GET OBSERVED DATA
1630  RETURN
2000  REM  *** FUNCTION CALCULATION ***
2500  REM  SUM OF SQUARES
2510 F = 0
2520  IF E$ = "A" THEN 2600
2530  IF E$ = "L" THEN 2700
2540  RETURN
2600  FOR I = 1 TO NO
2610 XP = XO(I): GOSUB 3000
2620 ER = YO(I) - YP:F = F + ER * ER
2630  NEXT
2640  RETURN
2700  FOR I = 1 TO NO
2710 XP = XO(I): GOSUB 3000
2720 ER =  LOG (YO(I) / YP):F = F + ER * ER
2730  NEXT
2740  RETURN
3000  REM  $$$ PREDICT YP FROM XP $$$
3999  RETURN
4000  REM  *** INITIALIZE ***
4800  GOSUB 5000: REM  SET UP MODEL PARAMETERS
4810  GOSUB 4900: REM  CHOOSE ERROR STRUCTURE
4820  GOSUB 6000: REM  GET OBSERVED DATA
4830  RETURN
4900  PRINT : PRINT "ERRORS ADDITIVE OR LOGARITHMIC?"
4910  INPUT "A OR L? ";E$
4920  IF  NOT (E$ = "A" OR E$ = "L") THEN 4900
4930  RETURN
5000  REM  $$$ SET UP MODEL PARAMETERS $$$
5999  RETURN
6000  REM  $$$ GET OBSERVATIONS $$$
6010 ER$ = "N": GOSUB 6100: IF ER$ = "Y" THEN 6010: REM  GET DATA
     , REPEATING UNTIL NO ERROR
6020  GOSUB 6500: REM  LIST DATA
6030  RETURN
6100  PRINT : PRINT "NAME OF  OBSERVED DATA FILE?"
6110  INPUT F1$
6120 D$ =  CHR$ (4)
6130  ONERR  GOTO 6800
6140  PRINT D$;"LOCK";F1$: REM  CHECK IF FILE EXISTS. IF NOT, BRA
```

```
NAME..TEMPLATE    DATE..03/13/81    TIME..1600    PAGE..02
STARTING LINE..0     ENDING LINE..63999

     NCH TO ERROR ROUTINE.
6150  PRINT D$;"OPEN";F1$
6160  PRINT D$;"READ";F1$
6200  INPUT 1T,N1$,N2$,NO,BY
6210  REM  ITEMS/REC.(2), ITEM NAMES, NO. OF RECS., AV. BYTES/REC
      .
6220  IF RN = 0 THEN  DIM XO(NO),YO(NO)
6230 RN = 1: REM  RUN CODE; DIMENSION STATEMENT BYPASSED IN LATER
      RUNS
6240  FOR I = 1 TO NO
6250  INPUT XO(I),YO(I)
6260  NEXT
6300  PRINT D$;"CLOSE";F1$
6310  POKE 216,0: REM  RESTORE ERROR FLAG
6320  RETURN
6500  PRINT : PRINT NO;" OBSERVATIONS READ FROM THE"
6510  PRINT "  TEXT FILE ";F1$;" ."
6520  PRINT "DO YOU WANT THEM LISTED ON THE"
6530  PRINT "  SCREEN, PRINTER, OR NEITHER?"
6540  INPUT "S, P OR N? ";RE$
6550  IF  NOT (RE$ = "S" OR RE$ = "P") THEN  RETURN
6560 SL = 1: REM  PRINTER SLOT
6570  IF RE$ = "P" THEN  PRINT D$;"PR#";SL
6600  PRINT : PRINT N1$,N2$
6610  FOR I = 1 TO NO
6620  PRINT XO(I),YO(I)
6630  NEXT
6640  PRINT D$;"PR#0"
6650  RETURN
6800 ER$ = "Y": REM  DEAL WITH ERROR
6810  POKE 216,0: REM  RESET ERROR FLAG
6820  POKE 768,104: POKE 769,168
6830  POKE 770,104: POKE 771,166
6840  POKE 772,223: POKE 773,154
6850  POKE 774,72: POKE 775,152
6860  POKE 776,72: POKE 777,96
6870  CALL 768: REM  RESET STACK
6900 CD =  PEEK (222): REM  ERROR CODE
6910  IF CD = 5 THEN  PRINT "ERROR: NOT ENOUGH DATA IN FILE"
6920  IF CD = 6 THEN  PRINT "ERROR: FILE NOT FOUND"
6930  IF CD = 13 THEN  PRINT "ERROR: N   A TEXT FILE"
6940  IF (CD = 5 OR CD = 6 OR CD = 13) THEN  RETURN
6950  PRINT "UNEXPECTED ERROR NO. ";CD
6960  STOP
7000  REM  *** FINAL OUTPUT ***
7999  RETURN
```

** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...TEMPLATE
DATE...03/13/81

BY
6200

CD
6900 6910 6920 6930 6940 6940 6940 6950

D$
6120 6140 6150 6160 6300 6570 6640

E$
2520 2530 4910 4920 4920

ER
2620 2620 2620 2720 2720 2720

ER$
6010 6010 6800

F
2510 2620 2620 2720 2720

FI$
6110 6140 6150 6160 6300 6510

I
2600 2610 2620 2700 2710 2720 6240 6250 6250 6610 6620 6620

IT
6200

N1$
6200 6600

N2$
6200 6600

NF
1100 1110 1120

NO
2600 2700 6200 6220 6220 6240 6500 6610

NP
1600 1610 1620

RE$
6540 6550 6550 6570

RN
6220 6230

SL
6560 6570

XO(
2610 2710 6220 6250 6620

XP
2610 2710

YO(
2620 2720 6220 6250 6620

YP
2620 2720

```
]
NAME..TRY   DATE..03/13/81   TIME..1600   PAGE..01
STARTING LINE..0    ENDING LINE..63999

1  REM  TRY DEC 22/81
2  REM  REV MAR 02/81
50  HOME : PRINT "PROGRAM TO TEST USER ROUTINE"
60  PRINT "  FOR MICRO SIMPLEX"
70  PRINT
90 B5$ =  CHR$ (7): REM  BELL
100  PRINT "TEST FOR PREDICTION OR"
110  PRINT "  FUNCTION CALCULATION?"
120  INPUT "P OR F? ";R5$
130  IF R5$ = "F" THEN 40000
140  IF R5$ = "P" THEN 41000
150  GOTO 120
200  REM *** PARAMETER SET-UP ***
210  PRINT : PRINT "WHAT IS THE NUMBER OF  PARAMETERS"
220  PRINT "  OR COORDINTES?"
230  INPUT "N = ";N
240  RETURN
300  REM  *** PARAMETER ENTRY ***
310  PRINT : PRINT "ENTER A TRIAL PARAMETER VECTOR:"
320  FOR I5 = 1 TO N: INPUT X(I5): NEXT
330  RETURN
40000  REM  *** FUNCTION CALCULATION ***
40010  GOSUB 200
40020  GOSUB 300
40030 NF = 0: GOSUB 1000
40040  PRINT B5$: REM  BELL
40050  PRINT "*** USER'S INITIALIZATION COMPLETED."
40100 NF = 1: GOSUB 1000
40110  PRINT B5$: REM  BELL
40120  PRINT "*** FUNCTION EVALUATION COMPLETED."
40130  PRINT "POINT:"
40140  FOR I5 = 1 TO N:V0(I5) = X(I5): NEXT
40150 W0 = 39: GOSUB 59500
40160  PRINT "FUNCTION VALUE: ";F
40200  PRINT : PRINT "PRESS ANY KEY TO CONTINUE . . .";
40210  GET R6$: PRINT
40220 NF =  - 1: GOSUB 1000
40230  PRINT B5$: REM  BELL
40240  PRINT "*** USER'S FINAL OUTPUT COMPLETED."
40300  GOSUB 300
40310  GOTO 40100
41000  REM  *** PREDICTION ***
41010 NP =  - 2: GOSUB 1500
41020  PRINT B5$: REM  BELL
41030  PRINT "*** OBSERVED DATA ENTERED."
41040  PRINT NO;" OBSERVATIONS"
41050  PRINT "FIRST:",X0(1);",";Y0(1)
41060  PRINT "LAST:",X0(NO);",";Y0(NO)
41100  PRINT : PRINT "LIST THEM ALL?"
41110  INPUT "Y OR N? ";R6$
41120  IF R6$ <  > "Y" THEN 41200
```

```
NAME..TRY   DATE..03/13/81   TIME..1600   PAGE..02
STARTING LINE..0    ENDING LINE..63999

41130  PRINT "OBS X","OBS Y"
41140  FOR I5 = 1 TO NO
41150  PRINT XO(I5),YO(I5)
41160  NEXT
41200 NP =  -- 1: GOSUB 1500
41210  PRINT B5$: REM  BELL
41220  PRINT "*** AUXILIARY MODEL PARAMETERS ENTERED."
41230  GOSUB 200
41240  GOSUB 300
41300  PRINT : PRINT "ENTER A VALUE OF X FOR PREDICTION:"
41310  INPUT "XP = ";XP
41320 NP = 1: GOSUB 1500
41330  PRINT B5$: REM  BELL
41340  PRINT "*** PREDICTION COMPLETE."
41350  PRINT : PRINT "AT X = ";XP;", Y = ";YP
41360  GOSUB 300
41370  GOTO 41300
59500  REM  VECTOR LIST
59510  REM  VECTOR=VO(N), PAGE COLS.=W0
59600 W5 = 0
59610  FOR I9 = 1 TO N
59620 W6 =  LEN ( STR$ (VO(I9))) + 3
59630  IF W5 + W6 > W0 THEN  PRINT :W5 = 0
59640 W5 = W5 + W6
59650  PRINT VO(I9);"   ";
59660  NEXT
59700  PRINT
59710  RETURN
```

```
   ** THE VARIABLE CROSS REFERENCE **

PROGRAM NAME...TRY
        DATE...03/13/81

B5$
90 40040 40110 40230 41020 41210 41330

F
40160

I5
320 320 40140 40140 40140 41140 41150 41150

I9
59610 59620 59650

N
230 320 40140 59610

NF
40030 40100 40220

NO
41040 41060 41060 41140

NP
41010 41200 41320

R5$
120 130 140

R6$
40210 41110 41120

VO(
40140 59620 59650

WO
40150 59630

W5
59600 59630 59630 59640 59640

W6
59620 59630 59640

X(
320 40140

XO(
41050 41060 41150

XP
41310 41350

YO(
41050 41060 41150

YP
41350
```